STUDIES ON LANGUAGE ANALYSIS PROCEDURE AND CHARACTER RECOGNITION(Dissertation_全文)

AUTHOR(S):

Nagao, Makoto

STUDIES ON LANGUAGE ANALYSIS PROCEDURE AND CHARACTER RECOGNITION

by

MAKOTO NAGAO

Doctoral Thesis, Kyoto University

December, 1965

STUDIES ON LANGUAGE ANALYSIS PROCEDURE AND CHARACTER RECOGNITION

by

MAKOTO NAGAO

Doctoral Thesis, Kyoto University

December, 1965

PREFACE

This thesis is a study of formal and natural languages and a study of character recognition, both of which constitute central problems in information processing.

Information processing having been progressed rapidly in recent few years is the development of the theories and techniques which perform very rapidly and mechanically the judgements and actions of high preciseness which human beings are doing ordinarily. The fundamental studies to do are the clarification of the mechanisms of man's thought processes, which is called "the study of automata or artificial intelligence"; the recognition mechanism of patterns, speech sounds, foreign languages etc., the ability of playing several games, problem solving etc., and the learning processes of human beings and so on.

An engineering fundamental for information processing is the digital computer. However to use the digital computer very easily the development of programming languages is imperative, for which many formal language theories have been studied. On the other hand natural language is one of the most interesting objects of man's activities to be studied in information processing field. Although the theories developed on formal languages (which are also called as artificial language against natural language) can be applied partly successfully on natural languages, it can be seen easily that it is

not enough. The studies on natural languages performed in engineering are mainly related to machine translation between different two or more languages, but it is still in the inchoate stage.

Pattern recognition function is another important activity of human beings. This can be said more primitive yet more higher functioning than the language activity. Pattern recognition includes the recognition of figures, (speech) sounds, etc. and even the recognition of languages. When the region of pattern recognition is restricted to characters only, the problem becomes rather easy and can be realized in engineering, which, if realized in simple machine, is very useful for actual use.

Pattern recognition may be said as the extraction of invariant parameters, but when patterns are very complicated and drastically different, the syntax of patterns becomes very important. The results of the language theory can be applied on this problem. On the other hand for the mechanical language processing the first thing is to recognize the written characters automatically by machine.

PART I of this thesis is a study of formal and natural languages especially from the syntactic and semantic aspects. The algorithms of the structural analysis are presented for a formal language, which do not require any other informations other than grammatical rules. This is a form directly applicable to the digital computer. Natural languages, on the other hand, have so many specialities, that it is very hard

to establish a definite grammar on it. For the structural analysis of natural languages the analysis algorithm obtained for the formal language is not enough. The semantic information of natural languages are imperative for the complete understanding of the properties of the languages. Therefore the semantics of the English language is studied from the standpoint of sentence generation.

All these results are directly applicable to machine translation of natural languages, which aims at the replacement of human translators in the confrontation with so many literatures to be interchanged among different nations.

PART II of this thesis is a study of character recognition machine. There are few character recognition machines developed in the U.S.A. and some other places. However they are all very expensive. This machine aims at the high recognition rate with very simple and inexpensive method. Pattern recognition is one of the most complex operations in human activities and the detailed mechanism of it has not yet been known. Even the most powerful digital computer of today is not enough for the processing and recognition of patterns. Therefore the study is confined itself in the recognition of numerals and English alphabet in typed form. However this kind of machine is very important for the practical use. Also in machine translation of natural languages by digital computer the automatic high speed input of original text is essential. In this respect the studies of PART I and PART II have close connection each other.

PART I of this thesis had an origin in the compiler construction of ALGOL 60, in which the author engaged five years ago under the guidance of Professor Takeshi Kiyono for the master's thesis in Kyoto University. After that period the author completed his study of formal language and applied that technique to the analysis and synthesis of natural language under the guidance and suggestion of Professor Toshiyuki Sakai.

The character recognition machine described in PART II of this thesis was completed three years ago under the direction of Professor T. Sakai and it was constructed by Nippon Electric Company Ltd. (NEC), Tokyo, about a year and a half ago. Other parts of PART II have been done during these two years in parallel with the construction of the recognition machine.

CONTENTS

PART II

DESIGN OF ASYNCHRONOUS CHARACTER RECOGNITION MACHINES

PART I

ANALYSIS PROCEDURES OF FORMAL AND NATURAL LANGUAGES

CHAPTER 1

INTRODUCTION

1.1 Problems in Formal Languages

By the recent rapid advances in the field of digital computer and information processing, many programming languages have been developed extensively in so many places. Among them FORTRAN[1] and ALGOL 60[2][3] are the most famous.

Whereas the system of FORTRAN is defined empirically, ALGOL 60 is defined very strictly by the language formation rules, i.e. by Backus normal form[4] which is essentially a phrase structure grammar.[5] The recent trend in the study of programming languages is to relate the language structure to the automata theories such as Turing machine on the one hand,[6][7] and to mechanize the compiler construction by the support of language theory on the other hand.[8][9] The theory of programming language is closely related to the study of formal language which has been developed in the field of logics.

Among lots of problems to be solved, an important problem for computer programmers is to determine whether a given string belongs to a language or not, and to know what is a structure of the string. By a programming language, the central problem is that a given string which may be a program is to be analyzed and to be converted into the form by which a computer can operate. A computer program which does this procedure automatically is a compiler. The construction of a compiler

needs a great amount of man power, and there have been very few theoretical supports for it.

In this thesis is shown a general mechanical method for the structural analysis of operator language which is more strict than context-free phrase structure language. As most part of ALGOL 60 has the structure of operator language, this structural analysis will contribute very much for the compiler construction.

Operator language [10] is defined as a phrase structure language, each rule of which has a special symbol, i.e. an operator. The concept of precedence of left and right sides is introduced for all the pairs of operators. This precedence relation is represented in a matrix which is derived automatically as a result of matrix operations among the matrices representing the grammar rules.

By the compiler construction at present the analysis procedure is found empirically from the grammar rules, so that it takes much time and is often accompanied by error. By the method presented here as compared to this, a table of grammar rules on which the compiler construction is based is obtained purely mechanically, and the process of the compiler can be written down in general form. Therefore even if a different grammar is given, only the grammar table is to be changed, while the algorithm of the structural analysis remains unaltered.

In a programming language by its nature, any ambiguity is not permitted both syntactically and semantically. This re-

stricts the structure of a language considerably. On the other hand ambiguity does exist very often in a general phrase structure language. The structural analysis in this case becomes rather complicated because several structures may correspond to a given string.

In this thesis a general method for the structural analysis of any context-free phrase structure language is presented, in which all the possible structures are obtained for a given string. Another peculiar feature of the analysis method is that a string is read in one by one from the left, at any stage of which all the possible structural analyses are offered.

This method of analysis is very powerful for the analyses of natural languages, in which an uttered sentence may often stop at any preferred point.

1.2 Problems in Natural Languages

On the foundation of the studies in formal languages, the author has had an interest in the similar process for natural languages.

Problems in natural languages, especially of machine translation may roughly be classified as the formalization of syntax of a language, the representation of semantics of words and phrases in a sentence, dictionary, and so on.[11] At present the most prevalent and potential method for the syntactic analysis is a phrase structure grammar, whose theoretical side has been studied extensively, and which is actually put into use in several MT research projects.

While on the other hand for the semantic side of natural language, there have been tried very few studies[12] and those are powerless for the application into the machine translation. Semantics has close connection to the human activities or real world as well as it has a variety of facets in the meaning itself.

Hitherto these problems of natural languages i.e. syntax and semantics have been studied independently each other, but the unified treatment of both problems is obviously required for the complete study of natural language. Standing on this view point the author has proposed a method of sentence generation by semantic concordance.[13][14] First a kernel sentence is generated by a phrase structure grammar, and then a proper transformation is applied to the kernel sentence. The first experiment of sentence generation was done by V. H. Yngve in 1961,[15] in which case only kernel sentences were produced without any consideration on the semantics of words and phrases. Therefore generated sentences were absurd or not understandable for us.

By the experiment the author has done, the efforts have been directed to the generation of meaningful sentences by taking into consideration the semantics of words and their mutual relationship in a sentence. Considering a natural sentence, we can point out a certain number of key words, the relation of which constitutes a meaning. A similar concept is introduced to the experiment of sentence generation. The method proposed in this thesis is to select first of all a

word which expresses the central concept of a sentence, which is usually a main verb of a sentence, and then to select a subject word and an object or a complement which do not contradict to the main verb as a sentence, and next some adjectival modifiers which also do not contradict to the subject or object in the meaning, and so on.

To do this each word must have proper semantic categories to which it belongs, these categories of words by which it can be modified, these categories of words with which it may have a subject predicate relation, a verb object relation etc. and so on. With these informations a sentence is generated. The quality of the generated sentences suggest the quality of structural rules and semantic information. Thus we can improve the grammar rules, semantic categories and their mutual relationship in a sentence. The research was done on the English language, in which a phrase structure grammar and transformational rules were established, several hundreds of English words were classified in semantic categories and were attached some additional semantic informations.

Moreover it was clarified that a structure to be explained by a phrase structure grammar is a phrase among whose elements there is a certain semantic relationship, while a structure to be explained by transformational rules is a phrase where it is improper to find a semantic relationship among the words constituting a phrase.

1.3 Outline of the PART I

The PART I of this thesis consists of two main subjects. One is the structural analysis of phrase structure grammars, the other is an experiment of sentence generation for the clarification of structure and semantics of natural languages.

Chapter 2 is a study of operator languages which are more restrictive than general phrase structure languages, but which make an essential part of programming languages. The concept of precedence for the left and the right is introduced and this precedence relation of all the pairs of operators is expressed in a matrix, which is shown to be derived from matrices representing the grammar rules. A general procedure of structural analysis of an operator language is shown with this precedence matrix of the operators of the language.

Chapter 3 is the study of structural analysis of a general context-free phrase structure language. First the canonical form of a context-free phrase structure language is defined, by which a formal representation of the tree structure is clarified. Next by using this expression all the possible structural analyses are produced for a given string, where the top symbol of an analyzed string may be any non-terminal symbol as well as the axiom. In this respect this method is more powerful than the previous ones which work only for the strings derived from the axiom.

Chapter 4 is devoted to the experiment of sentence generation which aims at a unified investigation of syntax and semantics. First the algorithm of the generation process is

explained, which includes the treatment of transformational rules and semantic information. The basic data are given of the semantic categories of words and their mutual relations in phrases. Next the position of morphophonemic rules and possible applications of the results of this experiments are discussed. Finally an experiment of the segmentation of Japanese sentences into word components is illustrated, in which semantics of words in a sentence is taken into consideration as well as their grammatical connectivities.

CHAPTER 2

OPERATOR LANGUAGES AND THEIR STRUCTURAL ANALYSES

2.1 Introduction

In this chapter the properties of operator languages are investigated.[16][17] A language means simply a set of strings in some finite set of symbols called the vocabulary of the language. A grammar is a set of rules that gives an enumeration of strings belonging to the language. Therefore a grammar of a language can be regarded as a tool of generation of sentences from the vocabulary of the language. Along this line the formal definition is given to a phrase structure grammar.

Programming languages such as ALGOL are defined by Backus normal form,[4][18] which is just the same as a context-free phrase structure grammar. Especially the greater part of a programming language is composed of an operator grammar which is more restrictive than general phrase structure grammar.

The study of the properties of operator language is important because the results of the study can be directly applied to the compiler construction of a programming language. A compiler might be regarded as an algorithm of structural analysis of the strings of a language.

2.2 Definition of Operator Grammars and Operator Precedence

First a phrase structure language is defined as follows. A language is generally a set of strings composed of a finite

set of symbols (or characters) called a vocabulary. A string consists of a finite number of symbols joined by concatenation from the vocabulary. Here we must be able to identify which strings belong to a language and which do not. Usually the number of strings of a language is infinite, so that the central problem is to define a potentially infinite set of strings over a finite vocabulary using a finite set of rules or instructions.

This finite rules which serve to define a language is called grammar rules or syntax of a language.

We define a language by a phrase structure grammar, which is presented as follows. The vocabulary of a language G is defined by

$$v(G) = \{\alpha, \beta, \gamma, \ldots . \}$$

A finite set Φ of rewriting rules ψ which are the following forms constitutes the grammar of a language.

$$\psi : \alpha \rightarrow \varphi$$

where α is a symbol of v(G) and φ is a string over v(G). These symbols which appear on the left side of $\rightarrow$ in rewriting rules ψ are called non-terminal symbols, while these symbols which do not appear on the left side are called terminal symbols. The rewriting rules ψ are applied as follows. If there exist strings u and v such that $y = u\alpha v$, and y belongs to a language G, then $z = u\varphi v$ also belongs to the language. In this case it is said that y directly produces z ($y \rightarrow z$), and conversely z directly reduces into y.

We define the case y produces z ($y \Rightarrow z$) and conversely

z reduces into y, if and only if there exists a sequence of strings $x_0, x_1, \ldots, x_n, (n \geq 0)$ such that $y = x_0$, $x_n = z$ and $x_{i-1} \rightarrow x_i$.

A phrase structure language is an ordered quadruple $G = (v(G), \Phi, T, A)$, where $v(G)$ is the vocabulary, Φ is the set of rewriting rules ψ, T is the set of terminal symbols, and is the subset of $v(G)$, while $v(G) - T$ is a set of non-terminal symbols, all of which occur as left part of at least one rule. A is the symbol which occurs in no right part of any rule of Φ and is called the axiom.

When there are many rules whose left parts are equal we will express these in the following way.

$$\beta = (\varphi_1, \varphi_2, \ldots, \varphi_m)$$

This is another expression of the set of rules

$$\begin{aligned} &\beta \rightarrow \varphi_1 \\ &\beta \rightarrow \varphi_2 \\ &\text{- - - - -} \\ &\beta \rightarrow \varphi_m \end{aligned}$$

So we will express a phrase structure grammar in the following way.

$$\left.\begin{aligned} \beta_1 &= (\varphi_{11}, \varphi_{12}, \ldots, \varphi_{1m_1}) \\ \beta_2 &= (\varphi_{21}, \varphi_{22}, \ldots, \varphi_{2m_2}) \\ &\ldots\ldots\ldots\ldots \\ \beta_\ell &= (\varphi_{\ell 1}, \varphi_{\ell 2}, \ldots, \varphi_{\ell m_\ell}) \end{aligned}\right\} \qquad (1)$$

where $S = (\beta_1, \beta_2, \ldots, \beta_\ell)$ is the set of non-terminal symbols $v(G) - T$.

Each φ_{ij} is called a syntactic unit and is the form,

$$\varphi_{ij} = \rho_{\nu_1} \rho_{\nu_2} \cdots\cdots \rho_{\nu_m} . \qquad (2)$$

where ρ_{ν_k} is a symbol of the vocabulary v(G). Each expression of (1) states that a non-terminal symbol on the left of equal sign can be replaced by any syntactic unit in the parentheses of the right.

Here the following two restrictions will be applied on the grammar rules ψ .

(i) In the expression (2),

$$\varphi_{ij} = /\delta/ ,^{*} \qquad \delta \in D \quad \text{for } m \geq 2.$$

(ii) The same δ appearing in a syntactic unit φ_{ij} can not be an element of any other syntactic units. That is to say δ is proper to only one syntactic unit.

From the restriction (ii) δ is a terminal symbol and D is the set of all the elements of δ .

The set D is a subset of the set of terminal symbol T. The set T - D is denoted as V. The phrase structure grammar thus defined is called grammar G and the language with this grammar is called an operator language $\mathcal{L}$.

Whether an operator language $\mathcal{L}$ has a recursive structure or not can be tested by the following procedure.(19)

$$\eta_0 = (\beta_{01} , \beta_{02} , \cdots\cdots , \beta_{0\ell}) = (0 , 0 , \cdots , 0)$$

$$\eta_1 = (\beta_{11} , \beta_{12} , \cdots\cdots , \beta_{1\ell}) = (\psi_1(\eta_0), \psi_2(\eta_0), \cdots, \psi_\ell(\eta_0))$$

In general at the i-th step:

$$\eta_i = (\beta_{i1} , \beta_{i2} , \cdots\cdots , \beta_{i\ell}) = (\psi_1(\eta_{i-1}), \psi_2(\eta_{i-2}), \cdots, \psi_\ell(\eta_{i-1}))$$

---

* /x/ means that the element x is a constituent of a syntactic unit.

where $\psi_k(\gamma_{i-1})$ represents the string set obtained from the substitution of $\beta_1, \beta_2, \ldots, \beta_\ell$ appearing in the right half of each equation (1) by $\beta_{i-1,1}, \beta_{i-1,2}, \ldots, \beta_{i-1,\ell}$. The substitution operation may in general continue infinitely. The recursiveness of the language $\mathcal{L}$ can be seen from the fact that there appears in the string set of γ_i, at least one terminal string in which the same characteristic symbol appears more than once. Hereafter we will assume that a language to be treated has no structural ambiguity.

We can introduce the concept of partial order relation to the set of $S \cup V$. When ρ can be found in a syntactic unit φ_{ij} belonging to β_i, where $/\rho/ = \varphi_{ij}$, $\rho \notin D$, then the following order is set up.

$$\beta_i < \rho$$

The ordering property in this sense is not satisfied generally in a phrase structure grammar, but the following fact remains because the grammar G is a set of sentence generation rules.

By deleting some proper syntactic units φ_{ij} from the grammar G, the set of $S \cup V$ can be made a partial order set. Here a phrase structure grammar G' is made from G by deleting the minimum number of syntactic units in order that G' can obtain the property of partial order. The grammar G' is called non-recursive grammar. This deletion operation can be done for any phrase structure grammar with the restriction that G can produce at least one terminal string. Thus we can write Hasse diagram for the elements of $S \cup V$.

As each element of D is proper to a syntactic unit φ_{ij}, the subset D_{β_i} of D can be defined as the elements of D appearing in φ_{ij} for j = 1, 2, ..., m_i. For the family of subsets ($D_{\beta_1}, D_{\beta_2}, \ldots\ldots, D_{\beta_\ell}$), we can introduce the order as the same order of ($\beta_1, \beta_2, \ldots\ldots, \beta_\ell$) defined above. This partial order of the set (D_{β_i}) has the following meaning.

In a terminal string where δ_i, δ_j appear,

(i) $\delta_i > \delta_j$ means that the syntactic unit containing δ_i is to be merged before that of δ_j.

(ii) When δ_i and δ_j can not be compared, it means that the both syntactic units containing δ_i and δ_j can be processed independently.

2.3 Structural Analysis of Non-recursive Operator Language and Pushdown Store

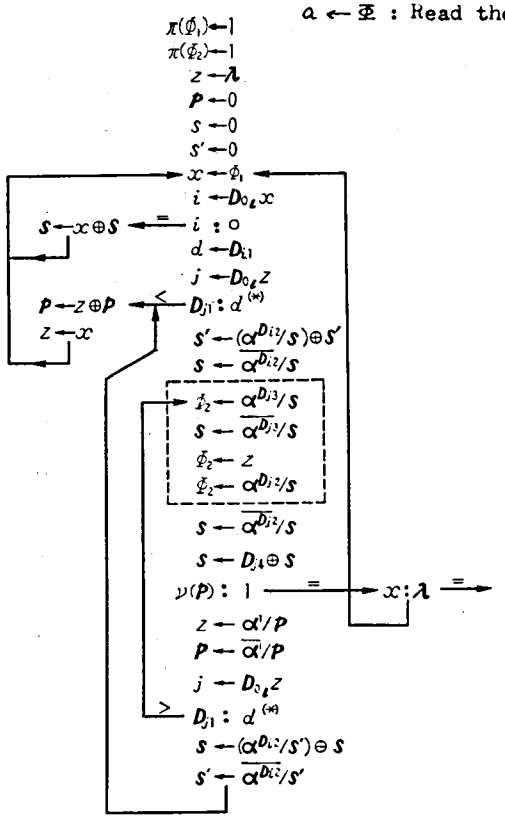
Each element of D is proper to a syntactic unit, and when the length of φ_{ij} is longer than 1, it has at least one element of D. So the syntactic analysis of a terminal string can be done depending exclusively on the precedence relation of characteristic symbols and no need to pay any attention to the other symbols.

Fig.2.1 shows the syntactic analysis of terminal strings generated by the grammar G'. The expressions in this figure are due to K. E. Iverson.[20] When a syntactic unit is such as

$$\varphi_i = / \delta_{i_1} / \delta_{i_2} /$$

then δ_{i_1} and δ_{i_2} has the same precedence and their treatment

$\nu(a)$: Number of components in vector a.
o : Null character.
$c \leftarrow b_i a$: $c = o$ if $a \notin b$, otherwise c is the j-origin index of the first occurrence of a in b.
$c \leftarrow u/a$: c is obtained from a by suppressing each a_i for which $u_i = o$ (compression).
$c \leftarrow a \oplus b$: $c = (a_1, a_2, \cdots a_{\nu(a)}, b_1, b_2, \cdots b_{\nu(b)})$ (catenation).
$w \leftarrow \alpha^j(k)$: First k of w_i are unity, where $k = \min(j,h)$, $h = \nu(w)$.
$\pi(\Phi) \leftarrow k$: Set file Φ to position h. Called rewind if $h = 1$.
$\Phi \leftarrow a$: Write a to the present position of file Φ and move to the next.
$a \leftarrow \Phi$: Read the present position of file Φ and move to the next.



Φ_1: Input file. A given terminal string is stored. It is assumed that the terminal partition λ is attached to the last of the string.

Φ_2: Output file.

P : Stack for the elements of D.

S : Stack for the elements of V.

S': Auxiliary stack for S.

D : Matrix of the grammar G'.

D_0: Column vector of $D \cup \lambda$.

D_1: Column vector indicating the partial order relation of the elements of D.

D_2: Column vector of $\nu(\Delta H)$, where $\Delta H \rho_i \Delta T = \varphi_{ij}$.

D_3: Column vector of $\nu(\Delta T)$, where $\Delta H \rho_i \Delta T = \varphi_{ij}$.

D_4: Column vector of β_j where $\rho_i \in D_{\beta j}$.

(*) : When D_{j1} and α can not be compared, the sequence is in the normal order.

Fig. 2.1 Algorithm of syntactic analysis for the language G'.

must be done by some other additional conditions such as the meaning attached to the unit. In Fig.2.1 the encircled part by a dotted line is that for the processing of a syntactic unit. This corresponds to the generation of an object program for the programming languages such as ALGOL, and to the word selection, word ordering and additional insertion etc. for the machine translation. In the figure the process is shown to transfer the syntactic unit to the output file. The element of D in the stack p is from top to bottom,

$$z > \delta_{p_1} > \delta_{p_2} > \cdots$$

so that when an element δ of D appearing after z has the relation $z \geq \delta$ or can not be compared with z, the precedence relation for z is locally maximum, and the syntactic unit containing z can be processed on the spot.

The working memory or the operand stack which can be used here is not the stack of last-in-first-out principle. As Fig. 2.2 indicates, the terminal symbols of V having a relation with the characteristic symbol z is stored in BC. When the next symbol δ of D has appeared, the operands CD which have a relation to δ but not to z might have been stored in the stack.

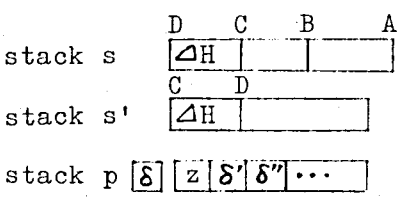


Fig. 2.2 Three stacks for the algorithm of Fig. 2.1.

Therefore for the operation of z, it is necessary that the elements CD are transferred to another auxiliary stack s' and

the elements BC come to the top of the stack. This situation arises when more than one non-terminal symbols are contiguously in a syntactic unit. On the contrary, when characteristic symbols and non-terminal symbols appear alternately in a syntactic unit, the above situation does not occur and the last-in-first-out stack can be used.

2.4 Explicitly Recursive Grammar

Grammar G', from its formation, can not produce infinite terminal strings. This is not preferable. So here a grammar having recursive structure is treated. A grammar G does not have the partial order relation that holds in a restricted grammar G' obtained from G. However it will be expected that G has the similar property.

The explicit recursion is here defined as a grammar containing rules of the form,

$$\beta_i \rightarrow \varphi_{ij} \quad \text{and} \quad \varphi_{ij} = / \beta_i /.$$

besides the rules of non-recursive grammar.

In this case the recursion is local for the non-terminal symbol β_i and does not have effects on the other part of the grammar, so that the partial order relation defined in the previous section holds as a whole. But the property of the characteristic symbols of β_i should be investigated further more. For the simplicity each syntactic unit belonging to β_i is restricted to have only one characteristic symbol.

$$\varphi_{ij} = / \delta /. \qquad \delta \in D_{\beta_i}$$

The following notations are defined. When a terminal

string is the form,

$$\Delta H_1 \cdot \delta \cdot \Delta H_2 \cdot \delta \cdot \Delta H_3$$

where ΔH_2 does not contain characteristic symbols,

δ on the left is denoted by ${}'\delta$, and

δ on the right is by δ'.

The precedence relation between these two ${}'\delta$, δ', is as follows.

(i) when $\varphi_{ij} = / \delta / \beta_i /$,

the successive replacement shows that

$$\varphi_{ij} = / \delta /(/ \delta / \beta_i /)/ = / {}'\delta / \delta' / \beta_i /,$$

from which ${}'\delta < \delta'$ is concluded.

(ii) when $\varphi_{ij} = / \beta_i / \delta /$

the successive replacement shows that ${}'\delta > \delta'$ is concluded.

(iii) when $\varphi_{ij} = / \delta / \beta / \beta' /$, $\varphi_{ij} = / \beta / \delta / \beta' /$, etc.,

the precedence relation between ${}'\delta$ and δ' can not be determined uniquely.

(iv) when two recursive rules are there for β_i, such that

$$\varphi_{ij} = / \delta_{i1} / \beta_i /, \qquad \varphi_{ik} = / \delta_{i2} / \beta_i /,$$

then ${}'\delta_{i1} < \delta'_{i2}$, ${}'\delta_{i2} < \delta'_{i1}$ are concluded.

However in more intricate rules as $\varphi_{ij} = / \delta_{i1} / \beta_i /$, $\varphi_{ik} = / \beta_i / \delta_{i2} /$, the precedence can not be determined uniquely. We can say thence that when the explicit recursion is of the form (i), (ii), (iv), the right and left precedence of δ can be introduced and the syntactic analysis can be done according to the precedence relations among the characteristic symbols alone.

For example the rule in ALGOL 60:

<simple arithmetic expression> ::= <term> | <simple arithmetic expression> <adding operator> <term>

indicates,

'<adding operator> > <adding operator>'

This means that the addition or subtraction should be done from left to right. Therefore a + b + c + d means (((a + b) + c) + d) definitely.

ALGOL 60 defines the left to right principle in semantics, but this will be unnecessary when the interpretation of a string follows strictly the syntactic analysis of the construction rules. The interpretation in semantics will help clarify the treatment of syntactic units such as $\varphi = \delta\beta\beta'$ etc., where it is not known which one of β and β' is to be treated first.

2.5 General Recursive Grammar and Precedence Matrices

A grammar has in general the rules of the form,

$$\beta_i \Rightarrow \varphi, \quad \varphi = /\beta_i/$$

but not $\beta_i \rightarrow \varphi$.

This is named as implicit recursion. The precedence relation among the elements of D varies according to their relative situation in a string.

Therefore the precedence relation for the right and left sides is to be clarified among all the elements of D.

The following properties of the terminal strings generated

by G are investigated here.

(A) What symbols of T constitute the terminal string of β_i ?

(B) What kind of symbols can be to the immediate right (or left) of an element ρ of T in a terminal string of β_i ?

(C) What kind of characteristic symbols can be to the nearest right (or left) of a characteristic symbol δ and in which precedence? In this case elements of V may be between the two characteristic symbols considered.

These problems can be solved by utilizing the Boolean matrices. The sum and product of two Boolean matrices are defined as follows.

$C = A \cup B$ means $Cij = Aij \cup Bij$.

$D = A \cap B$ means $Dij = \cup_k (Aik \cap Bkj)$.

Here, $\cup$ and $\cap$ are used in the usual sense.

(A) A matrix A is constructed from the grammar G in the following way. All the elements of $T \cup S$ are corresponded to the rows and columns of A (this is a square matrix). The matrix element Aij which refers to the row ρ_i, column ρ_j is defined as,

$Aii = 1$,

$Aij = 1$, for all (ρ_i, ρ_j), provided $\rho_i \in S$ and a syntactic unit φ belonging to ρ_i is the form

$$\varphi = /\rho_j/.$$

$Aij = 0$, otherwise.

Here, $A^2 = A \cap A$

$A^3 = A^2 \cap A$

. . . . . .

$A^n = A^{n-1} \cap A$,

are calculated successively and when a matrix A^K is obtained which satisfies $A^K = A^{K+1}$, this is denoted as A^∞. The matrix A^∞ shows the following property. When the elements belonging to the row β_i has value 1, the corresponding column elements can be components of β_i. This can easily be seen from the construction of matrix A and its production process.

(B) The Boolean matrix B having the same row and column elements as A is formed as follows.

Bii = 1,

Bij = 1, for all (ρ_i , ρ_j), provided $\rho_i \in S$ and a syntactic unit φ belonging to ρ_i has for its left-most element ρ_j ($\varphi = \rho_j/$),

Bij = 0, otherwise.

The Boolean matrix C is formed as follows

Cii = 1,

Cij = 1, for all (ρ_i , ρ_j), provided $\rho_i \in S$ and a syntactic unit φ belonging to ρ_i has for its right-most element ρ_j ($\varphi = /\rho_j$)

Cij = 0, otherwise.

Furthermore a matrix D of the following property is formed.

In a syntactic unit of the grammar (1) if $\varphi = /\rho_i \rho_j /$, then the element Dij corresponding to (ρ_i , ρ_j) is one (Dij = 1), otherwise zero (Dij = 0).

Then the following Boolean product is successively calculated.

$DB = D \cap B$

$DB^2 = DB \cap B$

. . . . . .

$DB^n = DB^{n-1} \cap B$

When $DB^K = DB^{K+1}$ can be obtained this is denoted by DB^∞ .

This matrix DB^∞ shows that when the element (i, j) of DB^∞ is 1, ρ_j can be juxtaposed immediately to the right of ρ_i with higher priority ($\rho_j < \rho_i$).

In the same way the matrix $D^t C^\infty$ is produced. This means that when (i, j) = 1, ρ_j can be juxtaposed immediately to the left of ρ_i with the higher priority ($\rho_j > \rho_i$).

From these two matrices,

$$E = DB^\infty \cup (D^t C^\infty)^t$$

is formed. This means that when (i, j) = 1, ρ_j can be contiguously juxtaposed to the right of ρ_i. These properties of terminal strings generated from the grammar G can be used for the checking of a given string if it can belong to that of grammar G.

Example:

$\delta_1 = (\uparrow)$, $\delta_2 = (\times, /)$, $\delta_3 = (+, -)$

$\delta_4 = (\,(\,)$, $\delta_5 = (\,)\,)$, $\alpha = (n, v)$

$$\beta_1 = (\,\alpha\,,\ \delta_4 \beta_4 \delta_5\,)$$

$$\beta_2 = (\,\beta_1\,,\ \beta_2 \delta_1 \beta_1\,)$$

$$\beta_3 = (\,\beta_2\,,\ \beta_3 \delta_2 \beta_2\,)$$

$$\beta_4 = (\,\beta_3\,,\ \beta_4 \delta_3 \beta_3\,)$$

The matrices of Fig.2.3 are obtained. DB^∞, for instance, shows that δ_4 can be juxtaposed to the immediate right of δ_1,

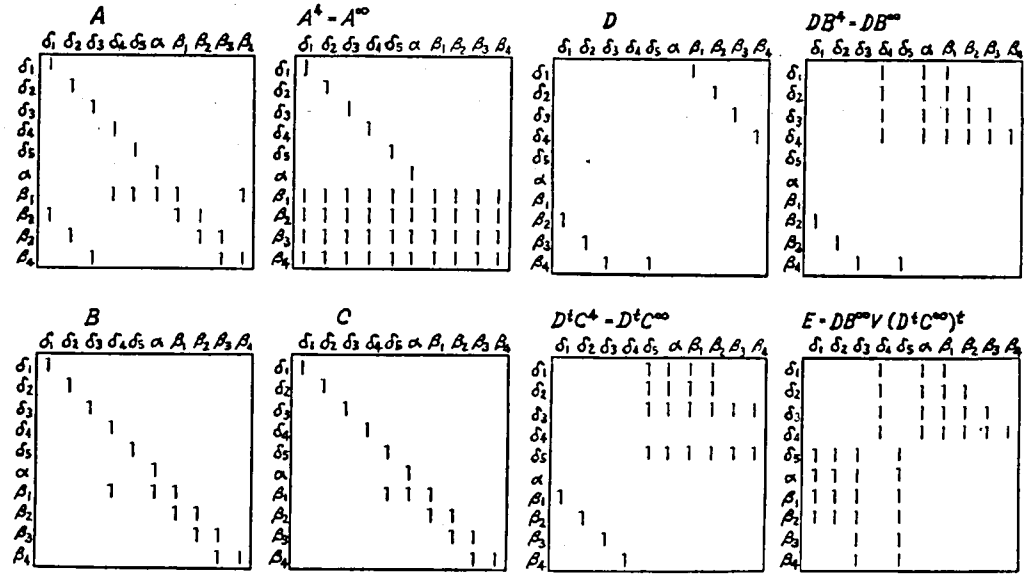


Fig. 2.3 Several matrices for the example.

$\delta_1 = (\uparrow)$, $\delta_2 = (\times, /)$, $\delta_3 = (+, -)$
$\delta_4 = (($, $\delta_5 =))$, $\alpha = (n, v)$

δ_2, δ_3, δ_4, and δ_4 has the priority. This corresponds to the sequences $\uparrow($, $\times($, $+($, $(($ and the part beginning with $($ should be processed before the operation of $\uparrow$, $\times$, $+$, $'($.

(C) The matrix F is constructed by applying the following procedure.

(i) When $\rho_i \in T$: $F_{\rho_i \rho_i} = 1$, $F_{\rho_i \rho_j} = 0$ $(i \neq j)$.

(ii) When $\rho_i \in S$:

a syntactic unit belonging to ρ_i is denoted as

$$\varphi_{ij} = \rho_{\nu_1} \rho_{\nu_2} \cdots \rho_{\nu_n}.$$

When $n = 1$, $F_{\rho_i \rho_{\nu_1}} = 1$, go to (vi).

When $n \neq 1$, supposing ρ_{ν_k} is the left-most element D and making $\ell = 1$, go to (v).

(iii) When $\rho_{\nu_\ell} \in S$:

if a terminal string of V can be generated which belongs to ρ_{ν_ℓ} ,

$F_{\rho_i \rho_{\nu_\ell}} = 1$, go to (iv).

If it can not be generated, $F_{\rho_i \rho_{\nu_\ell}} = 1$ and the repetition is terminated for ℓ , go to (vi).

(iv) $\ell + 1 \to \ell$ and if $\ell < k$ go back to (iii).

(v) If $\ell = k$, $F_{\rho_i \rho_{\nu_\ell}} = 1$ and the repetition for ℓ is terminated, go to (vi). If not go to (iii).

(vi) For all syntactic units of the grammar the above operation is done, go to (ii).

From this matrix F, the matrix DF^∞ is formed in the same way of (B). If in this matrix,

$$DF^\infty{}_{\rho_i \rho_j} = 1, \quad \rho_i, \rho_j \in D$$

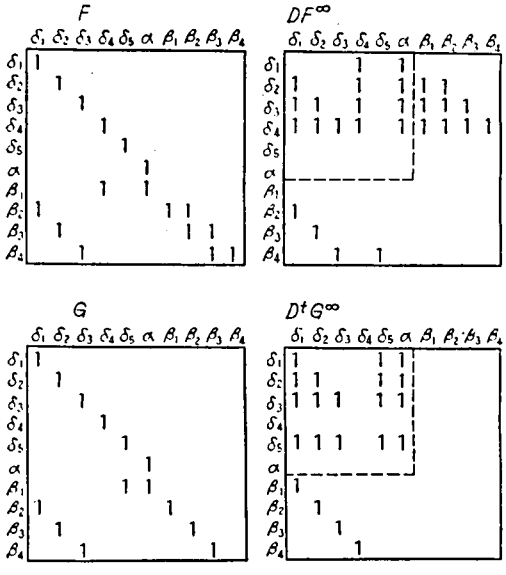
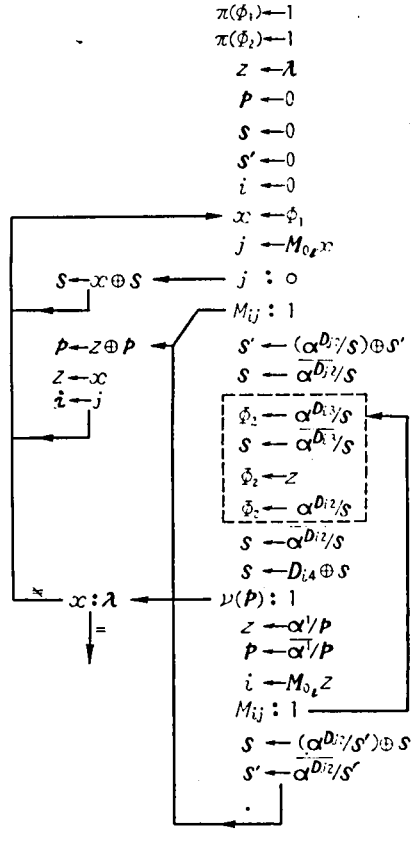


Fig. 2.4 Precedence matrices for the example.



M: Submatrix formed from DF^{∞} by deleting rows and columns coresponding to $S \cup V$ and adding the row of λ. The priority of λ is the lowest.

Fig. 2.5 Algorithm of general structural analysis with the precedence matrix M.

ρ_j can be to the immediate right of ρ_i (the elements of V can be between ρ_i and ρ_j), and the priority is $\rho_i < \rho_j$ (that is '$\rho_i < \rho_j$').

If the same procedure of (i)--(vi) is done with $\ell = n$ instead of $\ell = 1$, and $\ell - 1 \rightarrow \ell$ instead of $\ell + 1 \rightarrow \ell$, another matrix G is obtained. When DG^{∞} is calculated from D and G, this matrix means that if $DG^{\infty}_{\rho_i \rho_j} = 1$, $\rho_i, \rho_j \in D$, ρ_j can be to the immediate left of ρ_i (the elements of V can be between ρ_i and ρ_j), and the priority is $\rho_j > \rho_i$ (that is '$\rho_j > \rho_i$').

The matrices F, DF^{∞}, G, DG^{∞} for the above example are shown in Fig.2.4.

2.6 Structural Analysis of General Recursive Operator Language

Using these matrices DF^{∞} or DG^{∞} the syntactic analysis of the terminal strings generated by the grammar G can be done as shown in Fig.2.5, depending exclusively on the characteristic symbols. When a syntactic unit has more than one characteristic symbol, such as $\varphi = /\delta_1/\delta_2/$, the sequence of treatment among these symbols is to be determined semantically and these relations are to be added to the matrix DF^{∞}.

The property of a class of phrase structure language is explained, which has characteristic symbols in each syntactic unit. However the treatment developed in 2.5 and the resulting matrices do not depend on the characteristic symbols, so that they are effective for any phrase structure language. The mechanical analysis shown in Fig.2.5 can be constructed in hardware with the matrix of the grammar stored in a fixed mem-

ory.

2.7 ALGOL 60 and Polish Notation

The algorithmic language ALGOL 60 is the first programming language[21] which is defined strictly in the formal representation. But because of the universal nature of the definition there are many difficult problems for the construction of its compiler. In constructing compilers of arithmetic and Boolean expressions, Polish notation is used in many places.[22] Polish notation is the writing of algebraic or logical expressions which do not require grouping symbols and operator precedence conventions. For example, the expression A + B is written as AB + or +AB, and the expression,

$$A + (B - C \times D)/(E + F)$$

is written as,

$$ABCD\times - EF + /+$$

or

$$+A/-B\times CD + EF\ .$$

When an arithmetic expression is written like this, there is no need of the concept of operator precedence. By the expression ABCD × - EF + / -, the symbols are taken out one by one from left to right, until there appears the operator " × ". At this point the corresponding operation (multiplication) is performed with the two operands C and D, and the result replaces the three symbols CD× . Next the minus operation is done with B and the multiplication result of C and D, and so on.

This fact that no reference is needed for the operator

precedence in Polish notation can be explained by the theory presented in the above sections.(23)

Polish notation is an expression of the rules where each operator specific to a rule is placed to the right of the operands or to the left of the operands. In the latter case it is called inverse Polish notation. Therefore Polish notation constitutes an operator language.

Now the matrix D for the Polish notation is constructed as shown in Fig.2.6, where D denotes the operators, V denotes the operands, and S denotes the non-terminal symbols.

	D	V	S
D	0	0	0
V	*	*	*
S	*	*	*

Fig. 2.6 Matrix D for the rules of Polish notation.

	D	V	S
D	0	0	0
V	*	*	*
S	*	*	*

Fig. 2.7 Precedence matrix DF^{∞} for Polish notation.

Here all the elements of the rows of the operators D are zero, because there is no rule of the form that an element of V or S comes to the right of the operators. Therefore whatever F may be, DF^{∞} becomes as shown in Fig.2.7, where the rows of the operators D are zero.

This tells us that there is no operator which can be to the right of an operator with the higher precedence relation. This result thus guarantees the usual operations of Polish notation which does not require the operator precedence conventions.

CHAPTER 3

STRUCTURAL ANALYSIS OF GENERAL CONTEXT-FREE PHRASE STRUCTURE LANGUAGES

3.1 Introduction

In this chapter a method of structural analysis of general context-free phrase structure languages is presented. Generally languages under investigation have structural ambiguity, that is, a string of a language may have multiplicity of structure. If the number of grammatical rules is finite and a given string is also finite length, the structural analysis terminates in finite steps, and the string is determined whether it belongs to the language (in which case all the possible structures of the string are obtained simultaneously) or not. The method presented here[24] is superior to the previous ones[25] which give the structure for only the strings generated from the axiom, in the point that the structure can be clarified for any strings generated from any non-terminal symbols of the language.

Moreover all the structural analyses are obtained to the substring which is formed in the internal by the reading-in of one character by one from the left of the given string. Therefore even if a given string terminates in any place like conversational sentences[26] often end in the half, its structural analyses are obtained immediately. In this sense this structural analysis method may well suit to that of natural languages.

In section 2, the canonical form is defined for the context-free phrase structure grammar. This is that the right-half of a rewriting rule is composed of no more than two symbols. The procedure is shown to derive the canonical form of the grammar from an arbitrary context-free phrase structure grammar, and this canonical form is proved to be equivalent to the original grammar.

Sections 3 and 4 deal with a method of structural analysis by the canonical form and the representation of tree structure by a set of suffixes attacked to non-terminal symbols.

3.2 Canonical Form of Context-Free Phrase Structure Languages

We treat hereafter context-free phrase structure languages $G = (v(G), \Phi, T, A)$, whose rewriting rules ψ are of the form,

$$\beta \rightarrow \rho_1 \cdot \rho_2 \cdot \cdots \cdot \rho_n . \qquad n \geq 1$$

Here n is not less than 1, so that there is no rule of the form $\beta \rightarrow \Lambda$. Λ denotes the null string.

β is a non-terminal symbol and $\rho_1, \rho_2, \ldots, \rho_n$ are either non-terminal or terminal symbols. This set of non-terminal and terminal symbols is named vocabulary of a language G and is denoted by $v(G)$.

From a set of rewriting rules Φ we construct another set of rewriting rules Φ' in the following way.

(i) All the symbols of grammar G belongs to the vocabulary of G'.

$$v(G) \subset v(G')$$

(ii) The following rewriting rules of language G' are corresponded to a rewriting rule,

$$\beta \to \rho_1 \rho_2 \cdots\cdots \rho_n \qquad (n \geq 1) \qquad (1)$$

of language G.

(a) For $n = 1$ the rule $\beta \to \rho_1$ of language G itself is the rule of language G'.

(b) For $n = 2$ the rule $\beta \to \rho_1 \rho_2$ of G itself is the rule of language G'.

(c) For $n \geq 3$, the following set of rules belongs to grammar G'.

$$\left.\begin{array}{l} \beta_1 \to \rho_1 \rho_2 \\ \beta_2 \to \beta_1 \rho_3 \\ \cdots\cdots \\ \beta_{n-2} \to \beta_{n-3} \rho_{n-1} \\ \beta \to \beta_{n-2} \rho_n \end{array}\right\} \qquad (2)$$

Here $\beta_1, \beta_2, \ldots, \beta_{n-2}$ are different from any of the vocabulary v(G) so far as there is no rule in G whose right side of $\to$ is just the same as one of the rules (2). In the latter case the symbol on the left side of $\to$ should be the same as the one of the rule in G. These new symbols $\beta_1, \beta_2, \ldots$, of course, belong to v(G').

(d) Grammar G is defined only by (a), (b), and (c) mentioned above.

The rewriting rules thus obtained are called canonical form. The language $(v(G'), \Phi', T, A)$ thus defined is called language G'. Now we will show that the language G' defined as above is equivalent to language G. Here "equivalent" means

that all the terminal strings belonging to G are contained in the set of all the terminal strings of G' and vice versa.

We will show that a terminal string belonging to G belongs to G'. A terminal string of G is generated by a sequence of application of rewriting rules of G to the axiom A, so that it is enough to show a stage that a string generated by the application of a rule of G upon another string belonging to G can be generated by the application of some rules of G' in sequence to a string belonging to G', because the initial symbol is the same axiom A.

For a generation step

$$\chi\beta\phi \rightarrow \chi\rho_1\rho_2\cdots\rho_n\phi$$

of G, the following generation steps of G' are corresponded.

$$\begin{aligned}\chi\beta\phi &\rightarrow \chi\beta_{n-2}\rho_n\phi\\ &\rightarrow \chi\beta_{n-3}\rho_{n-1}\rho_n\phi\\ &\text{- - - - - - - - - -}\\ &\rightarrow \chi\beta_2\rho_3\cdots\rho_n\phi\\ &\rightarrow \chi\rho_1\rho_2\rho_3\cdots\rho_n\phi\end{aligned}$$

The axioms of G and G' are the same, so that if $\chi\beta\phi$ belongs to G, then it also belongs to G' by the above correspondence of generation. This proves that a string belonging to G also belongs to G'.

Next we will show that a terminal string belonging to G' also belongs to G. For any terminal string of G', there is corresponding tree structure S' which shows the generation process of the terminal string. A tree structure of S' is composed of rewriting rules which are either of the following cases.

(1) A rewriting rule of S' is the form of $\beta \to \rho$ or $\beta \to \rho\rho'$, and β, ρ, ρ' all belong to the vocabulary of G. In this case the rules $\beta \to \rho$ and $\beta \to \rho\rho'$ belong to Φ. Therefore if a string $\chi\beta\phi$ belongs to both G' and G, the generations $\chi\beta\phi \to \chi\rho\phi$ or $\chi\beta\phi \to \chi\rho\rho'\phi$ of G correspond to the generations of the same form of G'.

(2) A rewriting rule of S' is the form of $\beta \to \rho\rho_n$ and $\beta, \rho_n \in v(G)$, $\rho \notin v(G)$.

Because this rewriting rule must be the last rule of (2), there must be a rewriting rule of the form $\rho \to \rho'\rho_{n-1}$. If $\rho' \notin v(G)$ then the similar rule $\rho' \to \rho''\rho_{n-2}$, ... must exist, until finally $\rho^{(n)} \in v(G)$.

These rules are just the same as (2), because the left part symbols of (2) appear nowhere than only here. Thus the substructure of β of S' is composed of rewriting rules (2). When the rewriting rules (2) are applied to $\beta \to \rho\rho_n$ successively, until finally there remains no symbol which does not belong to v(G), the string obtained like this clearly belongs to language G. To summarize, if a string $\chi\beta\phi$ belongs to both G and G', the generation

$$\chi\beta\phi \to \chi\rho\rho_n\phi \to \chi\rho'\rho_{n-1}\rho_n\phi \to \cdots\cdots \to \chi\rho_1\rho_2\cdots\rho_n\phi$$

of G' is a unique expansion, to which the generation

$$\chi\beta\phi \to \chi\rho_1\rho_2\cdots\rho_n\phi$$

of G corresponds.

(3) A rewriting rule of S' is the form of $\beta \to \rho'\rho$ and $\rho \in v(G)$, $\beta, \rho' \notin v(G)$. This is a case of the intermediate stage of case (2). There must be a symbol $\rho'' \in v(G)$ to

the right of ρ of the substructure S', and

$$\beta' \rightarrow \beta\rho''$$

must exist. If $\beta' \in v(G)$, this is reduced to the case (2). If not, we can find another symbol $\rho''' \in v(G)$ to the right of ρ'' and

$$\beta'' \rightarrow \beta'\rho'''$$

must exist. In this way the process continues as far as the symbol to the left of $\rightarrow$, in this stage β'', belongs to v(G).

(4) A rewriting rule of S' is the form of $\beta \rightarrow \rho'\rho$, and $\beta \notin v(G)$, $\rho, \rho' \in v(G)$. This can be also reduced to the case (2) by the same reasoning as (3).

(5) The start point of generation, that is, the axiom, is the same in both languages.

From the above investigation if $\chi\beta\phi$ belongs to language G and G', $\chi\rho_1\rho_2\cdots\rho_n\phi$ belongs to G' and vice versa.

3.3 Inverse of Tree Structure

In this section the expression is defined of the connection of subtrees represented by the canonical form of the grammar. A non-terminal symbol ρ can be expanded in many ways. The structure of these subtrees from ρ is represented by attaching the suffixes to ρ. Proper sets T_i (i = 1,2,....) of non-terminals having the internal tree structures are set up, and their elements are numbered again properly 1,2,3,.... by natural numbers. This is necessary for the distinction of the same non-terminal symbol whose internal tree structures

are different.

When an element ρ_i of a set T_i and an element ρ_j of a set T_j are combined in a rewriting rule,

$$\beta \rightarrow \rho_i \rho_j$$

and β is an element of a set T_K, this relationship is expressed by the suffix attached to β. Generally several symbols which are the same non-terminal symbols, but which have different internal tree structures are included in a set T_i, so that natural number is attached to each ρ_i belonging to the set T_i.

Hereafter as the set T_i is constructively defined, a natural number, (the number of set T_i) +1, is assigned to a newly introduced element.

For the above example the following notation is used.

When $\rho_i(p',*,*,*,*) \in T_i$, $\rho_j(p,*,*,*,*) \in T_j$,

$\beta \rightarrow \rho_i \rho_j$ is expressed by

$$\beta(\nu(T_K) + 1, j, p, i, p') \in T_K.$$

Here $\nu(T_K)$ means the number of T_K. $*$ is used for proper entry which is not concerned here. This expression says that the p-th element of T_j and the p'-th element of T_i are connected as $\beta \rightarrow \rho_i \rho_j$, and β is $\nu(T_K)$ + 1-th element of T_K. These five indices tell the position of β in T_K and the positions of both elements in each set.

For the rules of the form $\beta \rightarrow \rho$, $\rho(p,*,*,*,*) \in T_j$, $\beta \in T_K$, β is denoted by

$$\beta(\nu(T_K) + 1, j, p, 0, 0) \in T_K.$$

Here as the object is to perform a structural analysis of a string, the set T_K is defined as composed of non-terminal

symbols which constitute the tops of trees inclusive of the k-th element and exclusive of the $k+1$-th element of a string counted from the left. Therefore j can be set equal to k ($j = k$), and the second parameter can be discarded. So the general form becomes

$$\rho(p,q,r,s) \in T_k .$$

This means that ρ is the p-th element of the set T_k and is composed of the s-th element of the set T_r (which is denoted as ρ_i), and the q-th element of the set T_k (which is denoted as ρ_j) in the rewriting rule

$$\rho \rightarrow \rho_i \rho_j .$$

From the construction algorithm of the set T_k, p is greater than q ($p > q$).

An example is shown in Fig.3.1. This tree structure is represented by the following expressions.

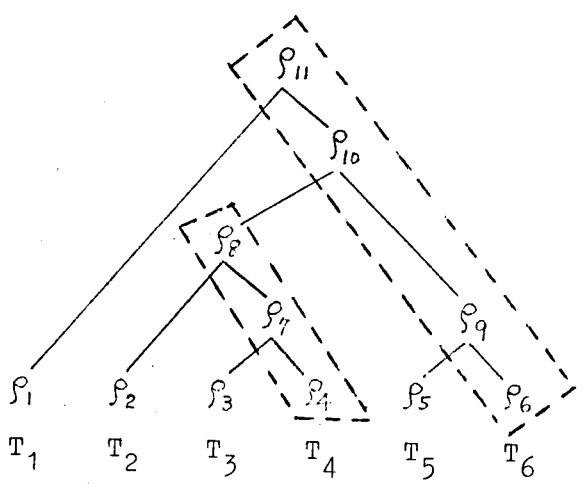


Fig. 3.1 An example of tree structure and the sets T_i.

ρ_7 is $\rho_7(2,1,3,1) \in T_4$

ρ_8 is $\rho_8(3,2,2,1) \in T_4$

ρ_9 is $\rho_9(2,1,5,1) \in T_6$

ρ_{10} is $\rho_{10}(3,2,4,3) \in T_6$

ρ_{11} is $\rho_{11}(4,3,1,1) \in T_6$

It may be understood from this example that the tree structure of the top element ρ_{10} has ρ_2 as the left-most element of its initial string.

This can be traced by the above expression such as, the left element of $\rho_{10}(3,2,4,3)$ is the 3rd element of T_4, i.e. $\rho_8(3,2,2,1)$, whose left element is the 1st element of T_2, i.e. ρ_2. If the left-most element of a terminal substring whose tree structure is derived from a non-terminal symbol ρ belongs to the set T_i, or in other word, is the i-th element from the left of the given terminal string, it is expressed as $i = Lm(\rho)$.

By the structural analysis of a given string, if $Lm(\rho) = 1$ for a certain non-terminal symbol ρ, the initially given substring from the top to the i-th element forms a syntactic unit ρ. Therefore for the given string $Z = X_1 X_2 \cdots\cdots X_n$, if $\rho \in T_{X_n}$ and $Lm(\rho) = 1$ then the given string Z is derived from the non-terminal symbol ρ, in other word, Z forms the syntactic unit ρ. The structure of Z from the top symbol ρ can be traced by the suffixes attached to each symbol.

3.4 Structural Analysis of General Context-Free Phrase Structure Languages

New sets S_α are defined in the following way for all the symbols α of vocabulary v(G') of language G' defined in 3.2.

(i) $\alpha \in S_\alpha$

(ii) $\beta \in S_\alpha$, when $\beta \to \rho$ and $\rho \in S_\alpha$.

(iii) $\Lambda \in S_\alpha$, when $\beta \to \rho$ and $\rho \in S_\alpha$ or $\beta \to \rho\rho'$ and $\rho \in S_\alpha$ (Λ is a null symbol)

(iv) For any terminal symbols α , $\Lambda \in S_\alpha$

The set S_α has the following properties.

(1) Each element of S_α produces the syntactic unit α itself, i.e. $\rho \Rightarrow \alpha$ holds for $\rho \in S_\alpha$.

(2) If S_α has null symbol, i.e. $\Lambda \in S_\alpha$, α may be a terminal symbol, or there is at least one rewriting rule of canonical form whose left symbol produces α.

(3) If $\Lambda \notin S_\alpha$, the rewriting rules containing α for its right part are only the forms $\beta \to \rho\alpha$.

Now the structural analysis of a given string $Z = X_1 X_2 \cdots X_n$ is done by picking up the terminal symbols X_1, X_2, $\cdots$ from left to right, and by performing the following sequence of operations.

(I) For X_1; $X_1(1,0,0,0) \in T_{X_1}$. X_1 is the first element of T_{X_1}.

For all $\rho \in S_{X_1}$ where S_ρ has Λ element, $\rho(\nu(T_{X_1}) + 1, 1, 0, 0) \in T_{X_1}$.

This means that $\rho \Rightarrow X_1$ holds. If $\Lambda \notin S_\rho$, $\beta \to \rho'\rho$ must hold for some β and ρ', but this is impossible because X_1 is the left-most element of a given string.

(II) For X_K ($K > 1$);

$X_K(1,0,0,0) \in T_{X_K}$, X_K is the first element of T_{X_K}.

(A) (i) For all $\rho \in S_{X_K}$ where $\Lambda \in S_\rho$; $\rho(\nu(T_{X_K}) + 1,1,0,0) \in T_{X_K}$

(ii) For all $\rho \in S_{X_K}$ where $\Lambda \notin S_\rho$;

If there are an element $\rho'(p',q',m',n') \in T_{X_{K-1}}$ and a rewriting rule $\beta \to \rho'\rho$, then $\rho(\nu(T_{X_K}) + 1,1,0,0) \in T_{X_K}$.

This means that a symbol $\rho \in S_{X_K}$ where $\Lambda \in S_\rho$ can exist as a non-terminal symbol, whereas a symbol $\rho \in S_{X_K}$ where $\Lambda \notin S_\rho$ must be merged to the form $\rho'\rho$, which can not exist, if there is no such $\rho' \in T_{X_{K-1}}$, and this denies the existence of ρ itself.

(B) (i) For $\rho(p,q,r,s) \in T_{X_K}$ where $\Lambda \in S_\rho$;

For all $\xi \in S_\rho, \xi \neq \Lambda, \xi \neq \rho$, where $\Lambda \in S_\xi$;

$\xi(\nu(T_{X_K}) + 1, p, 0, 0) \in T_{X_K}$.

For all $\xi \in S_\rho, \xi \neq \Lambda, \xi \neq \rho$, where $\Lambda \notin S_\xi$ and making $t = Lm(\xi)$,

if there are an element $\xi'(p', q', m', n') \in T_{X_{t-1}}$ and a rewriting rule $\beta \to \xi'\xi$, then $\xi(\nu(T_{X_K}) + 1, p, 0, 0) \in T_{X_K}$.

This means the same operation as A for the newly introduced elements at the stage (A).

(ii) For $\rho(p,q,r,s) \in T_{X_K}$ and making $t = Lm(\rho)$,

if there are an element $\rho'(p', q', m', n') \in T_{X_{t-1}}$ and a rewriting rule $\beta \to \rho'\rho$, then $\beta(\nu(T_{X_K}) + 1, p, t-1, p') \in T_{X_K}$.

This means the merge of two elements into one.

This procedure starts at X_1 and procedes to X_2, $X_3, \cdots, X_n$,

where the operations on X_K are recursively performed until there is no term which can be applied the operation.

The set T_{X_i} thus constructed by the read-in until X_i, contains all the non-terminal symbols which correspond to the structures of substring $X_1 \cdot X_2 \cdots X_i$.

In this way when the final terminal symbol X_n of the given string is read in and analyzed, the tree structure is corresponded to each of the element of T_{X_n} as its top symbol. Here the trees whose left-most elements are X_1, i.e. Lm = 1, are selected, whose top symbols represent the syntactic units of the given string, and these trees are the required structural analyses.

When the grammar has ambiguity, there may exist several structural analyses for a given string. In this case there are so many syntactic units of Lm = 1 in the set T_{X_n}.

Example:

A given sentence is:(26)

A man eating fish has an unbalanced diet.

The grammar rules are,

N $\rightarrow$ (man, fish, diet)

Det $\rightarrow$ (a, an)

Adj $\rightarrow$ (unbalanced)

Vt $\rightarrow$ (has, eat, unbalance)

N $\rightarrow$ $Adj_1 \cdot N$

N' $\rightarrow$ N

N' $\rightarrow$ Det $\cdot$ N

N' $\rightarrow$ N' $\cdot$ Adj_2

$Adj_1 \rightarrow N' \cdot Vting$

$Adj_2 \rightarrow Vting \cdot N'$

$Adj_2 \rightarrow Vted \cdot N'$

$Vp \rightarrow Vt \cdot N'$

$S \rightarrow N' \cdot Vp$

The sets S_α are in this case,

$Sn = (\Lambda, N, N')$ $\qquad$ $S_{Vt} = (\Lambda, Vt)$

$Sn' = (\Lambda, N')$ $\qquad$ $S_{Vp} = (Vp)$

$S_{Det} = (\Lambda, Det)$ $\qquad$ $S_{Vting} = (\Lambda, Vting)$

$S_{Adj_1} = (\Lambda, Adj_1)$ $\qquad$ $S_{Vted} = (\Lambda, Vted)$

$S_{Adj_2} = (Adj_2)$ $\qquad$ $S_S = (\Lambda, S)$

For the axiom, which does not appear in the right half of any rewriting rules, the set S is the axiom itself and null symbol Λ .

Each step of analysis of the given sentence by the given rules is in Table 3.1. In the set Tg , the elements whose Lm is equal to 1 are S(10), S(12), and S(13), and the structures for these analyses are shown in Fig.3.2. All these three analyses can actually exist, because we have such a sentence as, "A man eating fish called the piranha is found in the tropical waters of Brazil."

Table 3.1 Possible structural analysis of a sentence;

A man eating fish has an unbalanced diet.

word	set T_i	syntactic units belonging to the set T_i
A	T_1	Det(1,0,0,0)
man	T_2	N(1,0,0,0), N'(2,0,0,0), N'(3,1,1,1)
eating	T_3	Vting(1,0,0,0), Adj_1(2,1,2,2), Adj_1(3,1,2,3)
fish	T_4	N(1,0,0,0), N'(2,0,0,0), N(3,1,3,2), N(4,1,3,3), N'(5,1,3,2)
		N'(6,1,3,3), Adj_2(7,2,3,1), N'(8,7,2,2), N'(9,7,2,3), N'(10,3,1,1)
has	T_5	Vt(1,0,0,0)
an	T_6	Det(1,0,0,0)
unbalanced	T_7	Ved(1,0,0,0), Adj_1(2,0,0,0)
diet	T_8	N(1,0,0,0), N'(2,0,0,0), N(3,1,7,2), N'(4,1,7,2), Adj_2(5,2,7,1)
		N'(6,3,6,1), Vp(7,6,5,1), S(8,6,4,2), S(9,6-4,5), S(10,7,4,6)
		S(11,7,4,8), S(12,7,4,9), S(13,7,4,10)

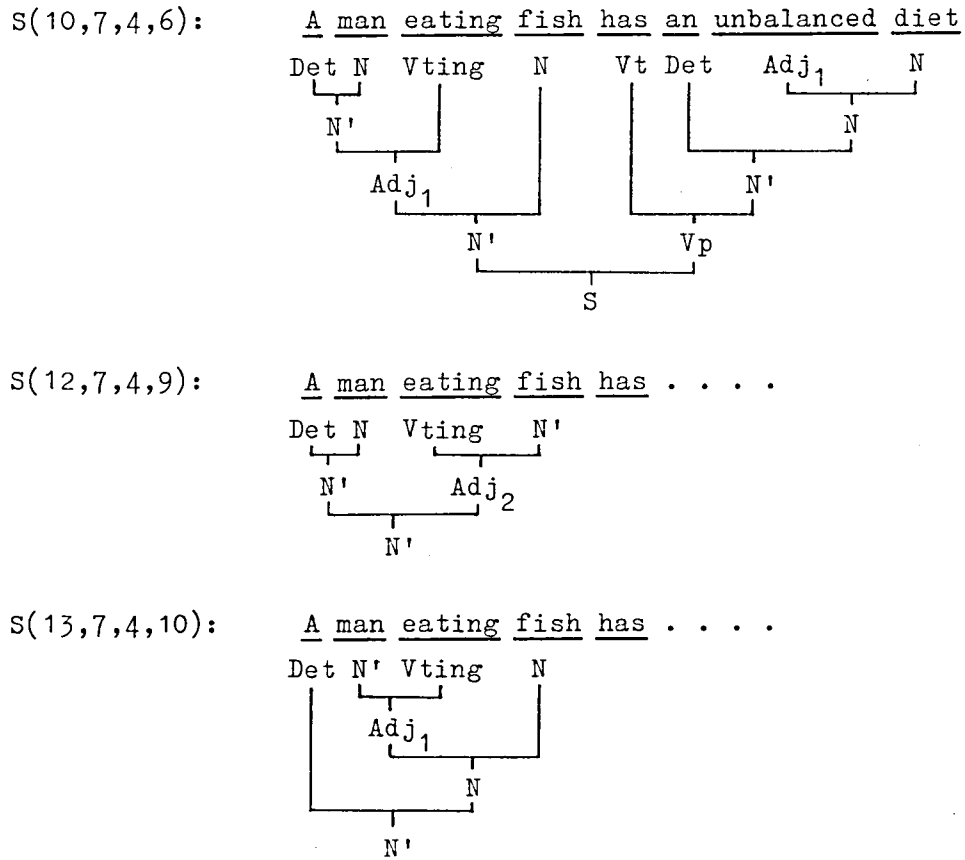


Fig. 3.2 Tree structure representations of;

A man eating fish has an unbalanced diet.

CHAPTER 4

SENTENCE GENERATION BY SEMANTIC CONCORDANCE

4.1 Problems in Machine Translation and Computational Linguistics

The possibility of using digital computer for the translation between two natural languages was suggested by A. D. Booth and some others in 1946[11] and some trials were done in word-for-word bases. After that time the rapid advances in computer technology has enabled the use of large memory capacity, and has made a project of machine translation (MT) a current one. Many MT projects have made it clear that the word-for-word translation is insufficient for the different language systems, and that the syntactic structures of a language must be taken into consideration.[27] The famous book, "Syntactic Structures" of N. Chomsky[28] which appeared in 1957, clarified that the sentence structure has three different levels, i.e. phrase structure grammar, transformational grammar, and morphophonemic rules. He also insisted that a grammar must generate all the sentences belonging to a language and no more.

The properties of the phrase structure grammar which N. Chomsky defined has been extensively studied in connection with the theory of automata and there have appeared many fruitful results.[6][7][29]

On the other hand actual experiments on real languages also were done in many places based on language theories. The conclusion has been, however, that the machine translation

is not possible at the present knowledge and technology.[30][31] The meaning is that this direction of study is very important, but that the multifarious facets of natural language are beyond our treatment.

We have no complete or enough tools for syntactic description, nor have we any methodology for the representation of semantics.

By the syntactic or structural description is meant the scope of morpheme, word, phrase, clause and sentence levels. At present in the machine translation study, morpheme and word levels are treated separately from others. This is mainly a dictionary problem and other levels from word to sentence are a problem of structural analysis.

The most famous and powerful theory of language is a phrase structure grammar. There are many variations to this when they are applied to language analysis. However the fundamental principle is to construct a higher level phrase from several adjoining elements of lower levels, so that it is also called immediate constituent method.[32] This grammar reflects our concepts of English grammar very well and is highly suited for the handling by digital computer. Many other analysis methods such as dependency analysis of the RAND Corp.[33] and predictive syntactic analyzer of NBS[34] and Harvard University[25] are proved to be equivalent to the phrase structure analysis.[6]

Next let's see the following famous example.[28]

Colorless green ideas sleep furiously.

This string (here the problem is whether this is a sentence or not, so we do not use "sentence" but we use "string") may not be regarded as a sentence of the English language and may be better excluded from the language, although this seems to be grammatical.

However the problem is what the criterion of exclusion is.

Another example is,

I bought a car with four dollars.

I bought a car with four wheels.

The difference of these two sentences is only dollars and wheels, but the sentence structures are completely different.

Here the problem is how to find this difference of sentence structures. These are the problems which can not be explained by the syntax, and are generally called the problems of semantics. It may be noticed that semantics is intrinsic in machine translation, because machine translation is essentially the search for the sentence of equivalent meaning or is the meaning transfer from one language to another.

In this way semantics is a central problem in language processing. However an approach to semantics is far more difficult than to syntax, because semantics has close relation to our intellectual activities or to the real world. This is why there are so few investigations and so few powerful results in the study of semantics of language.(35)(36)(37) The followings are some of the problems to be considered in semantics.

First the meanings of a word are to be represented in the

form applicable to digital computer. Each word bears the image of human culture so that many facets of the usage of a word must be investigated. For example "pigeon" is a kind of bird, can fly, is not harmful, and is related to an abstract noun "peace" at least for the Japanese people and so on. Those encyclopedic informations are to be suitably abstracted to a set of semantic categories.

Next problem is the relationship of two or more words when they appear in a sentence. A sentence expresses a relationship among words which appear in it. Their relationship can not be understood by the mere collection of the words, but the syntax existing among the words plays an important role. This is the semantic relationship in a phrase.

In the above example,

I bought a car with four dollars.

I bought a car with four wheels.

if the relationship of four words, bought, car, dollars, and wheels, are grasped correctly with the help of structural information, the structural analysis will be done successfully.

Concerning the meaning, P. M. Roget published his famous dictionary, "Thesaurus of English Words and Phrases" in 1852. (38) This was the classification of words according to the ideas that they express. Recently the rapid advances of digital computer technology have enabled statistical methods such as concordance which, for example, counts the cooccurrence of two specific words in a certain relation by digital computer. These methods, however if to be successful, need a great amount

of input data.

Dictionary is another important problem in machine translation. Hitherto this problem has not been thought so serious because they thought that the dictionary was only the entry of words. However the situation has changed so much that the dictionary is provided for words, conjugation of words, grammatical rules, semantic informations, the similar informations of target language and so on. That is, the dictionary plays a central role in machine translation. The size of a dictionary is another problem. Nowadays several ten thousands of dictionary entries are not rare. In this case the access time for a certain entry becomes an important factor in machine translation, and the updating of the dictionary information becomes a trouble.(39)

4.2 Methodology

At present on behalf of the experience obtained from the research already done, we have fairly exact knowledge about the real difficulties in the MT research which are to be solved in the near future. Among the problems the most important ones might be how to construct the syntax of a language and how to grasp the semantics of sentences of the language.

There have been many excellent contributions to the problems of syntax, but there are still few to the problems of semantics and the interrelationship between syntax and semantics. The author has tried an investigation in this area by the method of generation of English sentence.(13)(14)

The first paper ever published concerning the generation of sentences might have been that by Prof. V. H. Yngve of MIT in 1961.(15) Here the method has been adopted once again from the following points of view.

(i) It acts as a powerful test to the study of sentence structure.

(ii) It acts as a powerful test to the study of semantics.

(iii) It acts as a powerful test to the study of the relationship between syntax and semantics in natural language.

Generally speaking the sentence generation method, contrary to the analysis of a given sentence (which is guaranteed to be a correct one), tends to demand a severe construction of syntactic rules and word selection rules. It may seem at the present level of machine translation that the treatment of the sentence property in its entirety is too difficult to realize. But if we hope to have the translation as perfect as possible, we are necessarily to confront with this problem. The quality of a linguistic theory of a language may be best examined by the generation of sentences according to the linguistic theory.

Especially the effect of the interrelationship between syntax and semantics seems to be clearly exemplified by this so to speak "crude" test.

Thus the sentence generation method surely responds to this overall treatment of the sentence property. This may be considered as a step towards the general theory of natural language.

Several methods have been developed for the description

of sentence structure. The syntax of English is here represented by a phrase structure grammar, transformational grammar, and morphophonemic rules. The kernel sentence is generated by a phrase structure grammar, then some proper transformational rules are applied to it, and then the modification of the sentence by morphophonemic rules produces the final output.

These rules should generate "conceivable sentence structures", although the actually used sentence structures have several constraints. These are,for example,

(i) the depth of the sentence structure,

(ii) the coordination structure,

(iii) the intrinsic unsymmetry of sentence structure ----progressive structure, top heavy structure etc.

In general the rules which are suitable for analysis of a given sentence seem to differ from the rules which generate good sentences. The difference between these two is the difference between the actual spoken sentences and the conceivable sentences. Here we can see man's tendency to the language structure. Therefore it will not be worthless to know the frequency ratios of the phrases used in the actual sentences.

The next question, and the more difficult one than the former, is the determination of what is the proper meaningful sentence.

The test for the semantic anomaly of a sentence is far more difficult than the test for the grammaticality of the sentence. The following three levels of criteria might be supposed for the correct sentence.

(i) The grammatical sentence which is spoken or written by the average person (and the sentence which conveys a concrete concept without knowing the circumstances the sentence is spoken). Here "grammatical" covers the phonology, phonemics, morphology, syntax etc. These sentences which are grammatical but which are contradictory in meaning and which we do not speak are to be rejected.

(ii) The sentences which are incomplete in the word usages, inflexions and so on but which convey clear understandable concepts. These are the so-called corrigible sentences.

(iii) The sentences which are grammatical but carry no concrete meaning if they are not supplemented by tediously long explanations about the righteousness of the expression.

The second criterion was adopted for the generation of English sentences. That is because we can transform the corrigible sentences into the complete ones comparatively easily by checking the concordance of gender, number, case etc. Hereafter the main concern is therefore with the sentence which carries very definite concept, that is to say, the sentence of complete semantic consistency.

4.3 Sentence Generation by Semantic Concordance

The process of generation of the affirmative active declarative sentences is presented here by the phrase structure grammar. By the generation of the kernel sentence the attention is on the structural balance of the whole sentence, the

influence of the choice of a word to the other part of the phrase, and their relation to the unified concept of the sentence. An expansion rule has a main constituent and the other non-main constituents in the expanded part. The latter symbols may contain optional elements.

When an expansion rule is applied to a non-terminal symbol, to which there is already given a concrete word, the word is assigned to the main constituent of the expanded part. The words to the non-main constituent symbols are selected in relation to the main constituent word. A verb or a noun is taken as the main constituent of the non-terminal symbol "sentence" (initial symbol).

4.3.1 Terminology

A set of syntactic word classes (abbr. SWC)

$$S = (s_1, s_2, ...)$$

A set of words

$$W = (w_1, w_2, ...)$$

A set of semantic categories

$$P = (p_1, p_2, ...)$$

A set of non-terminal symbols

$$Z = (z_0, z_1, z_2, ...), \quad z_0: \text{axiom}$$

$$M = S \cup Z = (\varsigma_1, \varsigma_2, ...)$$

A set of p's belonging to a word w

$$P(w) = (p_1(w), p_2(w), ...)$$

$$P_i(w) = (p_{w_{i_1}}, P_{w_{i_2}}, ...)$$

A set of words belonging to a syntactic word class s

$$W(s) = (w_{S_1}, w_{S_2}, \ldots\ldots)$$

The type of expansion rules:

$$z \rightarrow \mathcal{X}, \quad z \in Z$$

$\mathcal{X}$: string of symbols in M. ($\mathcal{X}$ is called a syntactic unit)

$$\mathcal{X} = \rho_{x_1}\rho_{x_2}\cdots\rho_{x_n}.$$

$M(\mathcal{X}) = \rho_{x_m}$: main constituent of string $\mathcal{X}$.

$NM_j(\mathcal{X}) = \rho_{x_j}$: non-main constituent of $\mathcal{X}$. j is attached 1,2,3,... from left to right to the non-main constituents of string $\mathcal{X}$.

If $\mathcal{X}$ is composed of only one symbol, there is no non-main constituent.

Optional elements in the expansion rules are indicated by a pair of brackets attached to the symbols. An optional element in $\mathcal{X}$ can not be the main constituent.

The type of selection rules:

$s \rightarrow w$ or $s \leftrightarrow w$

Semi-terminal derivation:

Expansion rules are applied on non-terminal symbols, to the stage where there is no symbol to be expanded. The final string is composed of SWC.

A set of the derived main constituents for a symbol z:

A set of all the SWC which can be the main constituent of a non-terminal symbol z or the main constituents of a phrase which is generated by successive expansions of the main constituents of the original z.

$$S(z) = (s_{z_1}, s_{z_2}, \ldots\ldots\ldots)$$

$S(s_i) \equiv (s_i)$ is assumed.

4.3.2 The Process of Generation of a Kernel Sentence (I)

It is supposed that a sentence has one central thing or concept to be mentioned first of all. This is the main constituent of a sentence.

Then a second important concept is determined with its grammatical position, referring to the central concept already selected. Next a third important one is determined likewise, and so on. This process is formally represented in the following.

(i) $z_o \leftrightarrow w(s_i(z_o))$

$s_i(z_o)$ is an element of $S(z_o)$ which is the set of the derived main constituents for z_o. $w(s_i(z_o))$ is a word belonging to the set $W(s_i(z_o))$.

This shows the process starts from the selection of a word w for the axiom z_o, and the sentence is to be constructed with the core word w .

(ii) $z_o \rightarrow \mathcal{X}$, if $s_i(z_o) \in S(M(\mathcal{X}))$

The axiom z_o can be expanded into the syntactic unit $\mathcal{X}$ if and only if the already selected $s_i(z_o)$ at the stage (i) is contained in the set of the derived main constituents for $M(\mathcal{X})$.

(iii) $M(\mathcal{X}) \leftrightarrow w(s_i(z_o))$

The already selected word $w(s_i(z_o))$ is assigned to the main constituent $M(\mathcal{X})$ of the expanded syntactic unit $\mathcal{X}$.

(iv) $NM_k(\mathcal{X}) \leftrightarrow w_{m\ell k}(s_{\ell k}(NM_k(\mathcal{X})))$, for all k,

if a certain condition

$$f_{\mathcal{X}}(P(w), P(w_{m\ell_1}), P(w_{m'\ell'_2}), \cdots)$$

is satisfied.

To each non-main constituent $NM_K(\mathcal{X})$ is corresponded each word $w_{m\ell K}$ if the semantic categories for the words have a certain relation $f_{\mathcal{X}}$ with that of the word w assigned to $M(\mathcal{X})$.

At the n-th stage of the generation:

It is supposed that a word is already assigned to a non-terminal symbol z.

$$z \leftrightarrow w(s(z))$$

Then:

(i) $z \rightarrow \mathcal{X}$, if $s(z) \in S(M(\mathcal{X}))$

(ii) $M(\mathcal{X}) \leftrightarrow w(s(z))$

(iii) $NM_k(\mathcal{X}) \leftrightarrow w_{m\ell K}(s_{\ell K}(NM_K(\mathcal{X})))$, if a certain condition

$$f_{\mathcal{X}}(P(w), P(w_{m\ell_1}), P(w_{m'\ell'_2}), \cdots)$$

is satisfied.

4.3.3 Condition $f_{\mathcal{X}}$

To all the elements of the semantic categories $P = (p_1, p_2, \ldots)$, the semantic distances are supposed to be defined.

$$d_{ij} = d(p_i, p_j) \qquad i \neq j$$

$$d_{ii} = d(p_i, p_i) = 0$$

The condition f may be the following.

$$z \rightarrow \mathcal{X} \qquad (\equiv \rho_{\mathcal{X}_1}\rho_{\mathcal{X}_2}\cdots\rho_{\mathcal{X}_n})$$

$$M(\mathcal{X}) \leftrightarrow w(s(z))$$

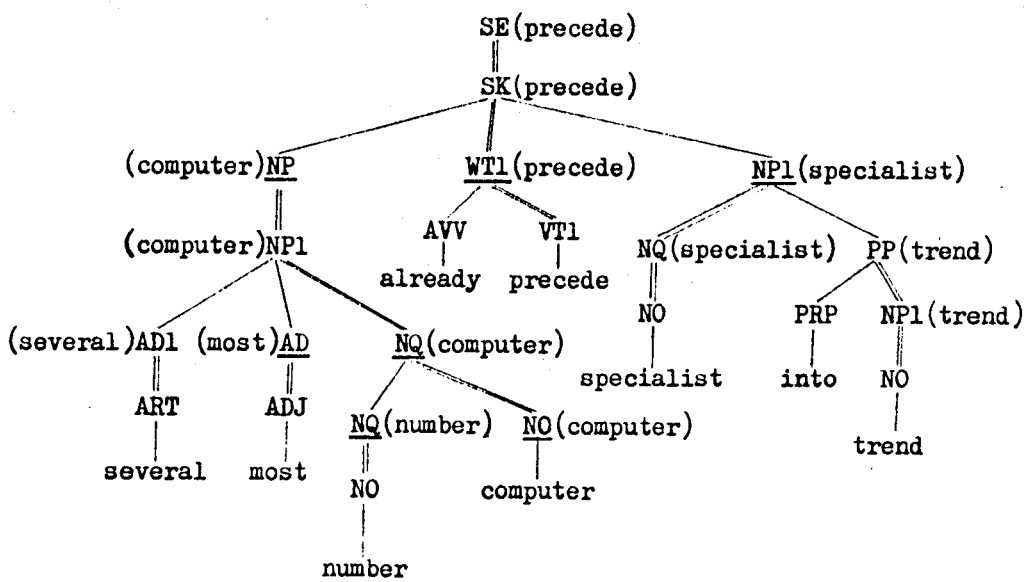
$$NM_K(\mathcal{X}) \leftrightarrow w(s(NM_K(\mathcal{X})))$$

$\beta_{\mathcal{X}} \leq \sum a_i d(p, p_k) + \sum b_{ij} d(p_i, p_j) \leq \alpha_{\mathcal{X}}$, $a_i, b_{ij}, \alpha_{\mathcal{X}}, \beta_{\mathcal{X}}$: constants i,j,k over all non-main constituent symbols of $\mathcal{X}$.

$$p_K \in P(w(s(NM_K(\mathcal{X}))))$$

$$p \in P(w(s(z)))$$

An example of this process is illustrated in Fig.4.1. The



Several most number computer already precede specialist into trend.

Fig. 4.1 Sentence generation from the axiom and a verb "precede".

double line indicates the main constituent of a phrase symbol which is written one line above. Certain semantic conditions are imposed on the pair of phrase names in phrases which are underlined.

4.3.4 The Process of Generation of a Kernel Sentence (II)

The generation process explained in 4.3.2 is from the axiom. But there are the cases where a sentence is to be constructed from an arbitrary grammatical position and a given word. For example when we write a complex sentence like "The old gentleman whom we saw at the theatre was his father.", the

main constituent of the subordinate clause is not "gentleman", but the verb "saw". So we must generate a sentence from a noun "gentleman" and its grammatical position: objective case.

The process is that first the start point of generation is given by a word and its part of speech in a sentence. Next a proper rewriting rule is selected which contains the part of speech of the word selected just now.

Then to the remaining elements of the rewritten phrase the proper words are assigned, the semantic categories of which coincide with the one of the already selected word. This process is continued as far as there remains no element which can be rewritten by a phrase. The process is formally represented in the following.

(i) Given z, w, s, where $s \leftrightarrow w$, $s \in S(z)$

(ii) A tree structure whose top symbol is z is constructed by the method explained in 4.3.2.

(iii) $z' \rightarrow \mathcal{X}$, $\mathcal{X} = \rho_{x_1} \rho_{x_2} \cdots z \cdots \rho_{x_n}$

A phrase z' is selected which contains z as a component of the expansion rule $z' \rightarrow \mathcal{X}$.

(iv) $z \leftrightarrow w$, $\rho_{x_i} \leftrightarrow w_{x_i}$ $i = 1,2, \ldots\ldots,n$

where certain condition

$$f_{\mathcal{X}}(P(w_{x_1}), P(w_{x_2}), \ldots, P(w), \ldots P(w_{x_n}))$$

is satisfied.

A proper word is corresponded to each phrase ρ_{x_i}.

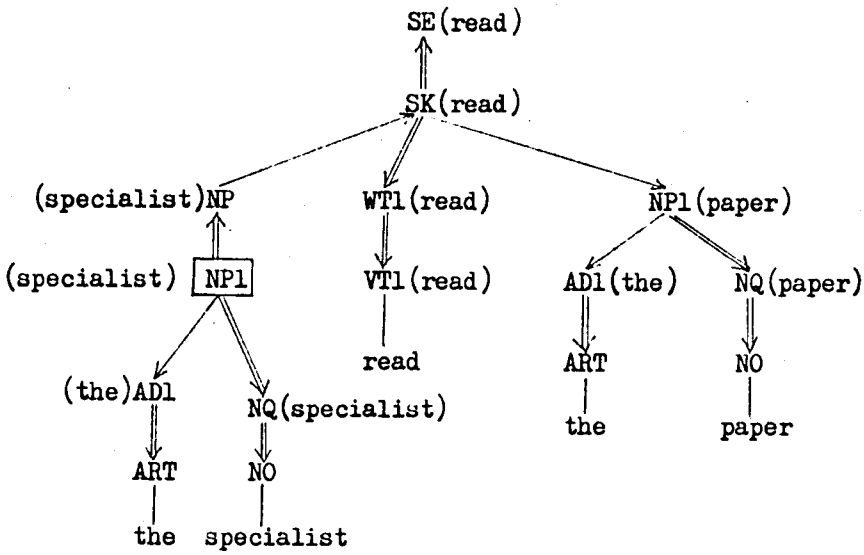
(v) Tree structures whose top symbols are ρ_{x_i} $(i = 1,2, \ldots)$ are constructed for all ρ_{x_i} by the method mentioned in 4.3.2.

(vi) At the stage (iii), z' is newly replaced by z and the

same operations from stage (iii) to (v) are performed.

(vii) When $z' = z_0$(axiom) is reached and the steps (iv) to (v) are completed, then the whole tree is accomplished under the axiom z_0.

An example of this process is illustrated in Fig.4.2.



The specialist read the paper.

Fig. 4.2 Sentence generation from NP1 and a noun "specialist". The direction indicates the steps the sentence is constructed.

4.4 Transformational Rules

4.4.1 Representation of the Rules

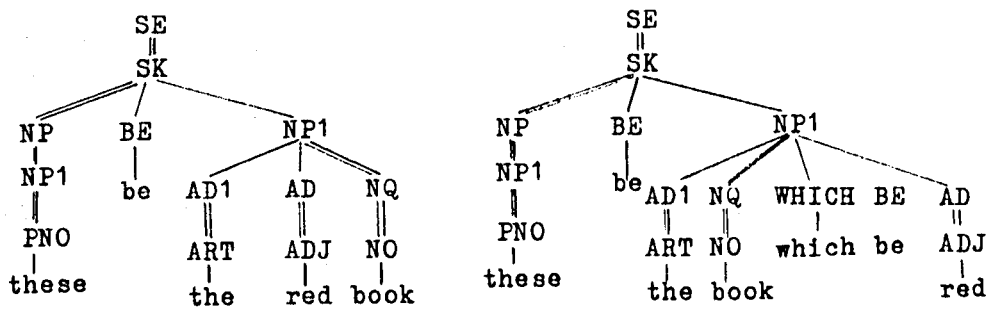
The transformational rules can explain many sentence structures which are difficult to be treated in an immediate constituent method. For example by a sentence "Is he young?", which is a question form of "He is young.", "is young" becomes discontinuous, separated by "he". This is difficult to treat by the immediate constituent method. It is far more natural

to explain this by the application of a transformational rule concerning question to the original affirmative sentence.

The transformational rules here used are classified to three types. The type 1 is unary transformations which may be thought of as converting a sentence from one to another. The type 2 is binary transformations which combine two sentences to form a third. And the type 3 is a transformation between two phrases. In all these cases, however, the transformational rules can be formally represented as follows.

$$z : X_1 \cdot X_2 \cdot \cdots \cdot X_n \rightarrow y_1 \cdot X_{i_1} \cdot y_2 \cdot X_{i_2} \cdot y_3 \cdot \cdots \cdot y_m \cdot X_{i_m} \cdot y_{m+1} \qquad (1)$$

The symbol z which is written on the left side means that this transformational rule should be applied to the phrase z. $X_1, X_2, \cdots, X_n$ are either the elements of M or words themselves. Among them may exists a special symbol $\emptyset$, which indicates that for this symbol $\emptyset$ there might or might not correspond some terms in an investigated phrase z. That is, $\emptyset$ expresses an arbitrary term. $y_1, y_2, \cdots, y_{m+1}$ are vacant, some symbols or words which are not equal to $X_1, X_2, \cdots, X_n$.



NP1 : $\emptyset \cdot AD \cdot NQ \rightarrow \emptyset \cdot NQ \cdot WHICH\ BE \cdot AD$

Fig. 4.3 Transformation in a phrase.

X_{i_1}, X_{i_2}, $\cdots$, X_{i_m} are some symbols among X_1, X_2, $\cdots$, X_n.

The phrase z has an internal tree structure, so the transformational rule is applied to this tree. An example of this is illustrated in Fig.4.3.

In this figure a noun phrase "the red books" is transformed to another noun phrase "the books which are red". This transformation is done by the rule,

NP1: $\emptyset$·AD·NQ $\rightarrow$ $\emptyset$·NQ·WHICH BE·AD

There are problems in the transformational rules such as follows.

(i) We have no definite criteria as to what kind of sentence structure is to be treated in the scope of phrase structure grammar, and what is in the scope of transformational rules.

(ii) We can name empirically or informally the transformational rules such as passive, "that" deletion, complement/object transposition, etc., but the formal representation of these rules in the form of (1) without contradiction for all the sentence structures generated from the specified phrase structure grammar, is difficult.

(iii) Transformations which accompany the changes in the part of speech or the morphophonemic forms of words are difficult to treat.

(iv) A transformational rule can not be applied unconditionally to the structure satisfying the rule form, but there are many cases where the application of rules depends on the semantics of the sentence.

4.4.2 Application of Transformational Rules

For the transformational rules of the type 1, a sentence is generated by a phrase structure grammar and at the same time the generation steps of the sentence are memorized by the tree structure representation. Next comes a transformational rule of the type 1 to this tree. If the rule is found to fit to the structure, then another tree is constructed from the orginal tree referring to the transformational rule.

Examples of this type are :

$\emptyset \cdot NP \cdot VT \cdot NP1 \cdot \emptyset \rightarrow 1 \cdot 4 \cdot BE \cdot 3 \cdot BY \cdot 2 \cdot 5$ (passive form)

This book emphasized the recent development clearly.

The recent development be emphasize(d) by this book clearly.

$\emptyset \cdot NP \cdot VI2 \cdot NP1 \cdot \emptyset \rightarrow 1 \cdot WHAT \cdot DO \cdot 2 \cdot 3 \cdot 5 \cdot ?$ (question)

Last year John became a doctor of philosophy at thirty.

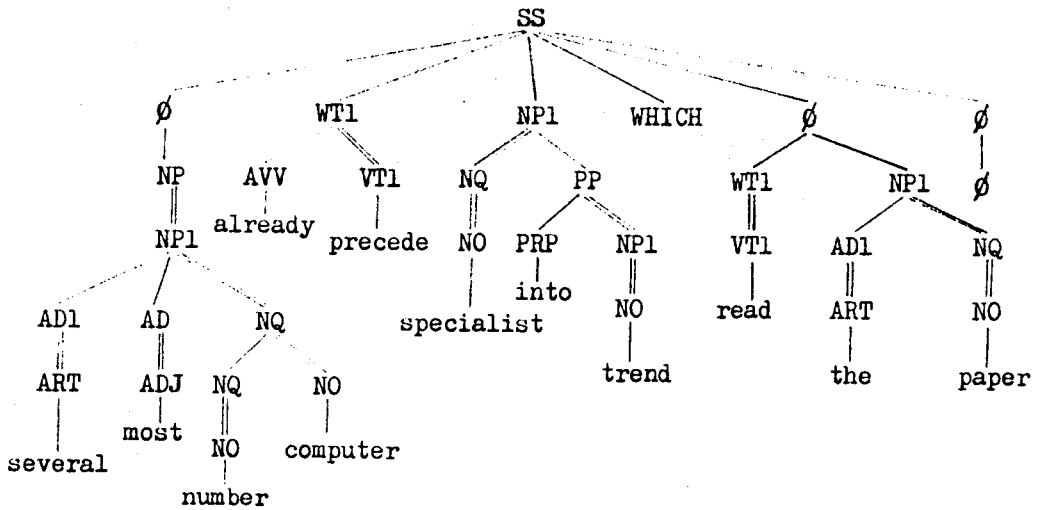
Last year what do John become at thirty?

The application of morphophonemic rules to these transformed sentences are explained in 4.6.

For the transformational rules of the type 2, first a sentence is generated by the phrase structure grammar, with its internal tree structure. Then a proper phrase name is selected which is a proper branch point of the tree, with the word attached to the phrase name. Next the generation of another sentence is done starting with the phrase name and the corresponding word, which are selected just now. This generation is by the method explained in 4.3.4.

Then the two sentences thus generated have a same word,

which is the key point in the usual transformational rules



Several most number computer already precede specialist into trend which read the paper.

SS : Ø·WT1·NP1·Ø·CM·NP1·Ø ⟶ 1·2·3·WHICH·7·4

Fig. 4.4 Transformation between two sentences. CM : comma

of the type 2.

An example of this type is illustrated in Fig.4.4. This is a combination of two sentences of Fig.4.1 and Fig.4.2. The rule applied here is,

SS: Ø·WT1·NP1·Ø·CM·NP1·Ø → 1·2·3·WHICH·7·4

and the generated sentence is,

Several most number computer already precede specialist into trend which read the paper.

This example indicates that a transformational rules can not be applied in every case, even if the structure satisfies the rule form. There are many other examples of this nature.

For the transformational rules of the type 3, mainly the noun phrases are studied which are the results of the applica-

tion of transformational rules to certain phrases, especially to the sentence form SE.

For example,

NP1·BE·NP1 ⟶ 1·CM·3·CM

Kennedy is the president of the U.S.

⟶ Kennedy, the president of the U.S.,

NP1·BE·PP ⟶ 1·3

Scientists are in the dome of the south pole

⟶ Scientists in the dome of the south pole.

This type of transformational rules are incorporated in the generation by the phrase structure grammar.

Another important transformational rules are those which accompany the change in the part of speech of words.

For example, nn : nominalization operator

VT1·NP1·∅ → nn·1·PRP·2·3

apply computer to the MT research

→ application of computer to the MT research

VT1·NP1·∅ → nn·1·BE GIVEN TO·2·3

consider the problems of the theory

→ consideration is given to the problems of the theory

This type of transformational rules remains to be investigated, in which the word dictionary should have information about the interchange of the parts of speech. Examples of these transformational rules are shown in Table 4.1.

Table 4.1 Examples of transformational rules.

Questions:

Ø NP BE Ø = 1 3 2 4 ?

Ø NP HV Ø = 1 3 2 4 ?

Ø NP MM Ø = 1 3 2 4 ?

Ø NP VE Ø = 1 DO 2 3 4 ?

Ø NP VC Ø = 1 DO 2 3 4 ?

Ø NP VO Ø = 1 DO 2 3 4 ?

NP VP = WT 2 ?

NR VQ = WT 2 ?

PN VQ = WO 2 ?

Ø VC NQ Ø = WT 1 2 4 ?

Ø VO NP OB Ø = WT 1 2 5 ?

Ø PO NP OB = 2 WT 1 ?

Negation:

NP BE Ø = 1 2 NT 3

NP HV Ø = 1 2 NT 3

NP MM Ø = 1 2 NT 3

NP VE Ø = 1 DO NT 2 3

NP VC Ø = 1 DO NT 2 3

NP VO Ø = 1 DO NT 2 3

There is(are):

NP BE AV = TR 2 1 3

NP BE AV = 2 TR 1 3 ?

NP HV BE AV = TR 2 3 1 4

NP HV BE AV = 2 TR 3 1 4 ?

Passive:

NP Ø VO NP OB Ø = 4 2 BE 3 6 BY 1 OB

Table 4.1 (continued)

Binary transformations:

NP ∅ VO NP OB ∅ CM SE = 1 2 3 6 TT 8

NP VP CM SE = 1 TT 4 2

NP VP CM SE = IF 4 OR NT 2

NP ∅ VO NP OB ∅ CM SE = 1 2 3 6 IF

NP ∅ VO NP OB ∅ CM SE = 1 2 3 6 HW

NP VP CM NP VP = 1 WH 5 2

NP ∅ NP OB ∅ CM NP VP = 1 2 5 3 OB WH 8

NP VP CM NP ∅ NP OB ∅ = 1 WH OB 4 5 8 2

NP ∅ VO NP OB ∅ CM NP ∅ BE NA ∅ = 1 2 3 6 8 OB 11 12

NP ∅ VO PN OB ∅ CM NP ∅ VO NP OB ∅ = 1 2 3 4 OB 6 8 OB 13

Where;

NP, NR, NQ are noun phrases,

PN is pronoun,

OB indicates the preceding term is objective case,

BE is be,

HV is have,

MM is auxiliary verb,

VE is intransitive verb, (complete)

VC is intransitive verb, (incomplete)

VO is transitive verb,

VQ is verb,

AV is adverb

CM is comma which separates two sentences,

SE is sentence, NT is not,

DO is do, TR is there,

WT is what, BY is by,

WO is who, TT is that,

HW is how, IF is if.

4.5 Semantic Categories and Their Relationship in Syntactic Unit

In the generation process thus defined, each word is determined by the selection rule $s \leftrightarrow w$ applied to SWC's. How this word selection should be done is the semantics here considered. If the selection is done randomly without any semantic restriction, completely anomalous sentence will appear.

To prevent this a new word is to be selected compatible with the already selected words which are in the neighborhood. Such semantic selection of words will especially be important in the syntactic relations such as

subject noun & predicate verb

subject noun & predicate verb & complement (or object)

adjective modifier & noun

noun & noun

adjective & adjective } (coordination structure)

verb & verb } etc.

Selection of a proper word in relation to the other words will eventually require the semantic notifications to the words and their mutual relationship in a sentence. In other words the system of semantic categories is to be set up and the meanings of all the words are to be represented in the system.

The construction of a system of the semantic categories may be done best by the replaceability relation among words in sentences. For example, to the verb "walk", there is a group of words which can be the subject to the verb "walk". To the word group thus formed, there will be another word group

which can be the predicate and has verb "walk" as its member. This word classification has not been tried yet on the whole scale, and indeed this is very difficult to do. So the author took a slightly different way, although the fundamental attitude of the word categorization was the replaceability of words in sentences.

It is postulated that all the words might be properly characterized by setting up a number of key concepts. For example a word "voyage" is categorized as journey with the additional images such as amusement, time duration, ocean etc. In fact when we speak we actually construct sentences fully aware of such additional meanings.

Thus the aim is to extract such word images and to know how these images are mutually connected in such and such sentence structures. So the extraction of semantic categories is done partly taking into consideration the Roget's thesaurus [38] and some other publications.[37][40] The following numbers are assigned to the semantic categories of the parts of speech.

100--299	verb
300--499	noun
500--699	adjective
700--799	adverb
900--	preposition

The ten's digit indicates the rough semantic categories in a part of speech and the one's digit is to the further classifications. At present the number of categories for the verb is

Table 4.2a Semantic Categories (verb)

Hand(100)

(111) close, open cover, fill, hold, drop, mark, plant, put, fire, take, draw, make

(112) keep, carry, use, have, get, give, help, bear, raise, hold, take

(113) connect

Eyes etc.(120)

(121) hear, find, see, look, watch

(122) say, speak, talk, tell, call, laugh, order, read, sing, state, cite, pronounce

Intelligent(130)

behaviour

(131) ask, answer, hear, find, order, cite, address

(132) add, get, receive, send, need, select, treat, eliminate, accept, arrange

(133) learn, read, find, write, see, process, apply, program, compute, specialize, compare, judge, indicate, understand, translate, know

(134) deal, treat

(135) interest, experience

Mental(140)

(141) feel, remember, think, reflect

(142) enjoy, thank, love, fear, like

(143) wish, hope, care, want

(144) stimulate, attract

Spiritual(160)

(161) believe, know, mind, think, mean

(162) attempt, aim, intend

Meals(170)

drink, eat

Social(180)

(181) build, found, publish, generate, develope, assemble, bridge

Table 4.2a (continued)

(182) kill, care
(183) pay, receive, buy, get, give, save, need, present, provide, precede, concede
(184) drive, fly, sail, ride, guide
(185) follow, lead, elect, participate
(186) live, work, make, cooperate, exist
(187) contribute, serve, help, save, pronounce
(188) consist, exist
(189) correspond, concern

Body action(190)

(191) play, show, try
(192) sit, stand, start, stop, put, set
(193) visit, call, meet, show, see, appear, address
(194) sleep, rest, remain, wait
(195) walk, move, pass, leave
(196) come, go, reach, run, stay, arrive

Change of state(230)

(231) change, turn, arise, remain
(232) start, go, come, drop, leave, begin
(233) extend, follow, increase, form

Natural phenomena(250)

(251) rain, blow, cover, drop
(252) burn, fire

(260)

(261) grow
(262) scatter

(280)

(281) be, become, seem
(282) need

Table 4.2b Semantic Categories (noun)

Human beings(300)

Table 4.2b (continued)

man	(301)	boy, child, girl, person, man, woman, Jack, Betty, Nelson
family	(302)	brother, company, family, father, friend, people, sister, son, mother
social	(303)	king, soldier, god, president
	(304)	specialist, reader, scientist, professional, group, blind, editor, debutant
	(305)	generation, group
Parts of 300(310)		
outward	(311)	body, ear, eye, face, foot, hand, head, hair
Animal(320)		
terrestrial	(321)	horse, lion
aerial	(322)	bird
aquatic	(323)	fish
Plant(330)		
	(331)	flower, grass
	(332)	tree, forest
	(333)	apple, pear
Nature(340)		
celestial	(341)	sun, moon, earth
atmospheric	(342)	rain, wind, air
geographic	(343)	river, hill, mountain, road, land, field, sea, Mt. Fuji, Kaatskill, Appalachia, Lake Biwa
minerals	(344)	rock, stone, gold, silver
water	(345)	water, sea, rain
Large things(360)		
movable	(361)	bus, train, car, ship
building	(362)	house, church, school, Kyoto Univ.
parts of 362	(363)	door, window, room
place	(364)	road, street, garden
Articles(370)		
books	(371)	book, picture, paper, story, Newsweek, Bible, handbook, library, monograph,

Table 4.2b (continued)

		article, summary, literature, supplement, journal, proceeding, report, volume, textbook
foods	(372)	food, egg, bread, milk, corn, salt, pepper, water, beer
furniture	(373)	table, box, bed, dress
playthings	(374)	ball, tennis
	(375)	processor, machine, computer
Mental action	(380)	
thinking	(381)	reason, idea, hope, mind, thought
feeling	(382)	love, life, fear
	(383)	aim, end
	(384)	readiness
	(385)	gratitude, thank, patience, acknowledgement
	(386)	knowledge, thought, view, opinion, reference, aspect, conception, comment, consideration, understanding
	(387)	sense, art, view
Action	(390)	
	(391)	life, love
	(392)	war
	(393)	question, answer, speech, call, order, judgement, citation, problem, example
	(394)	death, existence
	(395)	visit, watch, indication, advance, access, response, change
	(396)	work, help, treatment
	(397)	voyage, play, trip
	(398)	research, work, study, aid, contribution
	(399)	selection, processing, use, application, programming, elimination, addition, presentation, publication, specialization
Abstract	(400)	

Table 4.2b (continued)

(401) name, word
(402) thing, matter, something, state
(403) way, form, mean, point
(404) case, matter, measure, course, use
(405) cause, change, end
(406) color, sight
(407) sound, voice

(410)

(411) group
(412) gap, point, boundary, link
(413) interest, value, validity
(414) limitation, difficulty, success, problem
(415) kind, respect, branch, feature

Social terms(420)

(421) money
(422) bank, company, shop
(423) law, right
(424) city, country, street, town
(425) world, country, Japan, America, nation

(430)

(431) data, language, word, German, English, French, reference
(432) technique, science, manner, way, methodology, originality
(433) particular, detail, summary, assembly
(434) translation, generation, development, appearance, comparison, arrangement, cooperation
(435) information, fact, source, consequence
(436) subject, material, list, listing, title

Measure(440)

(441) one, two, three
(442) first, second, third
(443) feet, mile, length

Table 4.2b (continued)

(444) east, west
(445) right, left
(446) part, whole
(447) line, mark, point, top

Time(460)

(461) morning, night, day, hour
(462) today, day, week, Sunday, Monday
(463) spring, summer, winter, May, month
(464) date, period, time
(465) present-day
(466) year, generation

(470)

(471) field, area, circle, center
(472) course, state, live, manner, way, stage
(473) context, implication, content, indication, source

Table 4.2c Semantic Categories (adjective)

Quantitative(510)

number	(511)	all, many, several
quantity	(512)	all, much, little
cardinal	(516)	four, five, three
ordinal	(517)	first, last
multiplicative	(518)	half

Pronominal(520)

definite	(521)	same
indefinite	(522)	another, any, certain, other
distributive	(523)	every, each,
quantitative	(524)	any, various
	(525)	own

Outward state(530)

color	(531)	black, blue, green, red, white
shape	(532)	round, plain
length	(533)	long, short
height	(534)	deep, high, low
extent	(535)	wide, narrow

Table 4.2c (continued)

	size	(536) large, little, small, big, least
		(537) single, individual, numerous, multiple
Subjective(550)		
	beauty	(551) beautiful, pretty
	fair	(552) fair, clear, fine
	free	(553) free, fresh
	full	(554) full, complete
		(555) undisputed
Internal(570)		
	material	(571) gold
	weight	(572) light, heavy
	hardness	(573) hard, soft
	temperature	(574) cold, warm, hot
	new, old	(575) new, old, fresh
	soft	(576) soft, sweet, fresh
Human(580)		
	age	(581) old, young, fresh
	health	(582) sound, sick, dead, strong
Social state(590)		
	wealth	(591) rich, poor, cheep, modest
	position	(592) great, deep
	fact	(593) true, correct, natural
	restriction	(594) free, strict, major
		(595) skilled, detailed
		(596) commercial, available
		(597) neighboring
Mental state(600)		
	sentiment	(601) glad, happy, sad
	intimacy	(602) dear, ready, desirable, familiar
		(603) aware, wary, afraid
		(604) active, attractive
		(605) proud
Time(620)		
		(621) fast, quick

Table 4.2c (continued)

(622) ready

(623) late, recent, current, present, final

Valuation(640)

(641) good, bad, right, modest, valuable

(642) very, immense

(643) natural

Abstruct(660) (661) general, collective, common, particular, special, comprehensive, standard, specific

(662) next, certain

(663) present, absent, conventional

(664) related, informed, concerned

(665) complex, detailed, bound, complete

Relation(670)

(671) right, left

(672) close, near, intermediate

(673) direct, conclusive, extensive

(674) individual

(675) possible, impossible, necessary, enough

(680)

(681) digital, linguistic, automatic, spectral, technical, scientific

(682) literary, professional

(683) world-wide

Table 4.2d Semantic Categories (adverb)

Time(710)

(711) before, ago, early, lately, already

(712) today, soon, now

time duration (713) always, daily

(714) again, just

Place(720)

here, there, far, near, elsewhere

Number(730)

once, sometimes, often

Way(740)

well, thus, kindly, wisely, due, together

Table 4.2d (continued)

Degree(750)	little, more, greatly, enough, further, rather, somewhat
(760)	almost, yet, certainly
(770)	closely, highly, selectively, directly, primarily, successfully, clearly, probably, completely, widely, absolutely, similary
(780)	namely, mainly
(790)	nevertheless, hence, now, therefore, also

Table 4.2e Semantic Categories (Preposition)

Time(950)	about, after, around, at, by, behind, from, on, off, on, till, within, until
Means & object(960)	at, by, for, of, on, to, with, without
Place(970)	about, across, along, around, at, among, between, by, behind, from, in, under, off, on, upon, over, within, above
Movement(980)	across, after, along, from, into, off, over, to, against
Relation(990)	about, after, among, between, by, for, in, of, under, on, over, to, with, without, above, against

about 40, for the nouns about 90, for the adjectives about 50, etc. Different number is attached to each preposition. These semantic categories are shown in Table 4.2. Next the connectivity of words in a sentence is to be clarified. This is achieved by attaching several kinds of semantic categories to each word in the following way.

Verb:

P1: categories intrinsic to the verb.

P2: categories which can modify the verb (additional images of the verb)

P3: categories which can stand as subject to the verb.

P4: categories which can stand as object or complement to the verb.

P5: special prepositions following the verb if any.

P6: grammatical indication as to the form of the verb.

Noun:

P1: categories intrinsic to the noun.

P2: categories which can modify the noun (additional images of the noun)

P3: grammatical indication as to the form of the noun.

Adjective:

P1: categories intrinsic to the adjective.

P2: categories which can modify the adjective (additional images of the adjective)

P3: prepositions following the predicative adjective if any.

Adverb:

P1: categories intrinsic to the adverb.

This expresses, for example, that a verb can take a noun for the subject whose semantic categories P1 belong to P3 of the verb, and can take a noun for the object or complement whose semantic categories P1 belong to P4 of the verb, etc. These are the conditions f_x introduced in 4.3.3.

Examples of words having these connectivity informations are shown in Table 4.3.

The generation process is thus first the selection of a verb, and then the determination of subject, object or complement referring to the semantic concordance mentioned here. Therefore each expansion rule of the phrase structure grammar has the indication such as

SE → NP/P1,P3/·VT1·NP1 / P1,P4 /·PRP / P1,P5/·NP1

NP → ADJ / P1,P2/·NQ

In this example VT1, NQ which are underlined are the main constituents to which some concrete words are selected. So for NP, NP1, PRP and ADJ, proper words are to be assigned having the semantic concordance between the pair of categories bracketed by / / . The first element in / /is the category of the phrase to the left of / /, and the second element is that of the main constituent. Therefore a word is assigned to NP, whose semantic categories P1 have the same term in P3 of VT1, and so on.

But there are many grammatical phrases where it is not clear what kind of semantic relationship is to be established. The semantic relationship can not be attached clearly to the

Table 4.3 Semantic categories attached to words

Verb

add	P1 : 132		P5 : 980
	P2 : 730 760	build	P1 : 181
	P3 : 300		P3 : 300
	P4 : 369 420		P4 : 346 361 364 368
	P5 : 990	hear	P1 : 121 131
answer	P1 : 131		P3 : 300 320
	P2 : 730 740		P4 : 390 380 400 412
	P3 : 300		413 414 418
	P4 : 300 393 395 396	help	P1 : 112 186
bear	P1 : 112		P3 : 300 341 342 360
	P2 : 722 741		380
	P3 : 300 320 340 360		P4 : 300 320 380 390
	P4 : 381 400		400
	P5 : 960	see	P1 : 193 121 150
begin	P1 : 232 260		P3 : 300 320
	P3 : 300		P4 : 300 310 320 330
	P4 : 390		340 360 380
believe	P1 : 160	think	P1 : 160 130
	P2 : 743 744		P3 : 300 425
	P3 : 300		P4 : 380 390 402 403
	P4 : 300 380 400		405 425
bring	P1 : 196	enjoy	P1 : 142
	P3 : 300 320 341 361		P3 : 300 320 425
	380		P4 : 340 372 391 395
	P4 : 300		397 463

Noun

answer P1 : 393
P2 : 510 520 550 590 640 660

apple P1 : 330
P2 : 510 520 531 532 536 551 575

bank P1 : 368
P2 : 536 575 591 592 640 660

bird P1 : 322
P2 : 510 520 530 551 578

book P1 : 362
P2 : 510 520 530 551 572 641

change P1 : 413
P2 : 520 536 550 590 620 640

country P1 : 384 385
P2 : 510 520 536 551 590

flower P1 : 330
P2 : 531 536 551 576 591

hour P1 : 410
P2 : 597 594 620 640 660

order P1 : 393
P2 : 536 553 573 621 640 660

fear P1 : 391 395
P2 : 534 536 578 592 593

love P1 : 382 383
P2 : 550 574 576 578 593 595

right P1 : 388 411
P2 : 554 592 593 594 640

sight P1 : 405
P2 : 534 535 551 574 597 660

Adjective

old P1 : 575 581 623

flesh P1 : 575 576 553

strong P1 : 573 582

great P1 : 592

plain P1 : 532 596

free P1 : 553 594

right P1 : 538 593 641

phrases like,

main verb : adverbial phrases preceded by preposition

main sentences : subordinate clause

noun : its adjectival clause etc.

It is also difficult to find out the semantic relationship between the nouns of the form,

NO + NO, NO + NO + NO, NO of NO

such as,

machine translation, information processing machine

generation of sentences.

However when these phrases are given from suitable transformations of another phrases such as

solution of a problem ↔ solve a problem,

generation of sentences ↔ generate sentences,

the semantic relationship can be established to these two nouns in the phrase before the transformation is applied. In this case it is necessary to know the noun form of a verb or its vice versa. This information is contained in the word dictionary as P6 for the verb and as P3 for the noun.

The semantic relationship here introduced is essentially the connectivity between two words in a phrase, so there remains a possibility of generating absurd sentences. To prevent this, investigation is far more required on the minute mutual influence of meanings among words in a sentence.

4.6 Morphophonemic Rules

The author proposed that the morphophonemic rules can be

represented by a kind of operators operating on the words in the neighborhood. In it are included negation action, tense, case etc. The operators such as follows are studied.

n (not), pr (present tense), ps (past tense), inf (infinitive), sg (third person singular), pl (plural), sub (subjective case), obj (objective case) ing (ing form of a verb), ed (past participle) nn (nominalization of a verb), etc.

The functions of these operators are,

n + verb → do + not + verb

n + aux. verb → aux. verb + not

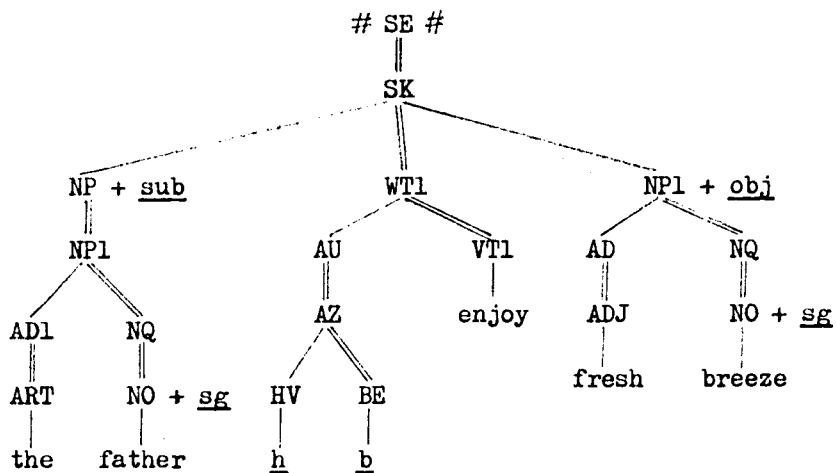
pr + verb → verb (present form)

ps + verb → verb (past form)

ing + verb → verb (gerund or present participle)

ed + verb → verb (past participle)

Besides these, for the verb BE(b) and HAVE(h), the following three steps are to be applied in this order.



the father sg sub h b enjoy fresh breeze sg obj

Fig. 4.5 Application of morphophonemic rules as operators.

(1) b + b + verb → be + being + ed + verb

(2) b + verb → be + ing + verb

(3) h + be → have been

For example in the generation of a sentence shown in Fig.4.5, the sentence obtained initially is,

the father sg sub h b enjoy fresh breeze sg obj

Here the following operations are applied.

sg obj → obj sg, sg# → #

noun obj → noun, b enjoy → be + ing + enjoy

ing + enjoy → enjoying, h + be → have been

sg sub → sub sg, sg have → has

noun sub → noun

And the final form is,

the father has been enjoying fresh breeze

These operators can appear both in the phrase structure grammar and in the transformational rules, but the operations of these are supposed to be done after the application of transformational rules. But there occur many complicated situations for the sentences after the application of transformational rules and to what extent this might be successful remains to be seen in the future.

4.7 Programming Techniques and the Generation Results

4.7.1 Programming Techniques

There are many phrase structure rules which are defined recursively so that there is a possibility that the same structure is repeated many times by the generation. Therefore the

adoption of a rule by random generation is to be made in proportion to the actual frequency of the appearance of the rule. For the rules of phrase structure grammar the following adoption ratios are attached.

$$z \longrightarrow \psi_1 \qquad (p_1)$$
$$z \longrightarrow \psi_2 \qquad (p_2)$$
$$\cdots\cdots\cdots$$
$$z \longrightarrow \psi_n \qquad (p_n)$$

This means that the expansion of a non-terminal symbol z to ψ_1 occurs by p_1%, to ψ_2 by p_2%, ..., to ψ_n by p_n%.

The sentence generation according to the algorithm 4.3.2 can be realized easily by a push-down-pop-up store or a stack. First the axiom is put in the stack. Next the first expansion rule of the axiom is adopted randomly. The axiom in the stack is canceled and on it the expansion rule is piled up.

Then the top element of the stack is examined whether it is a terminal symbol or not. If it is, it is printed out. If not an expansion rule of the symbol is randomly selected and is piled up in the stack, after cancelling the top symbol. In this way only the top symbol is expanded until there remains no symbol in the stack. This is the end of an expansion and the output is a generated sentence.

In the case of the sentence generation according to 4.3.4. the stack in this sense can not be used. The memory utilization is more complex.

The tree structure is to be memorized with the non-terminal symbol and a corresponding word for each node. For the appli-

cation of transformational rules to this tree structure and the rearrangement of the structure, the linkage of each non-terminal symbol in the tree is to be notified. In this way the three items i.e. non-terminal symbol, the corresponding word, and the linking address to the next expansion subtree are

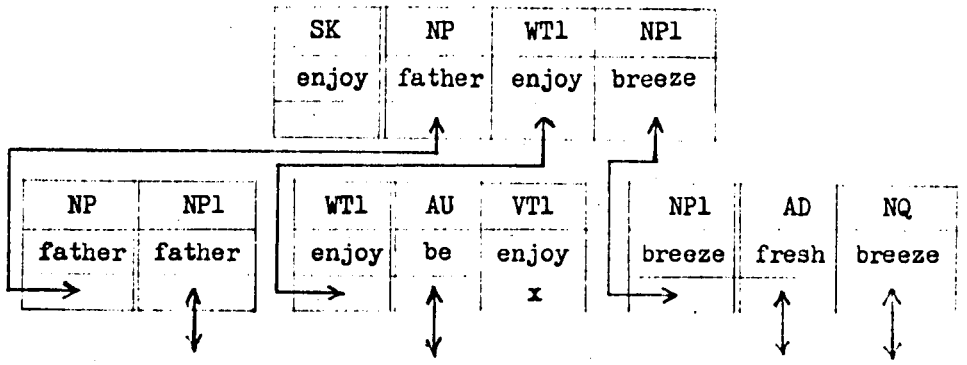


Fig. 4.6 Tree structure representation in computer memory.
x means no further linkage from here.

to be held in each node point. An example is shown in Fig. 4.6. This is a kind of list structure, but this is different from others in the point of reversible linkage among list elements.

4.7.2 The Generation Results

The experiment was done by a computer installed in Kyoto University, which is called KDC-I. It took usually 1 to 10 minutes to generate a sentence. This is because the computer does many false trials in the adoption of expansion rules and words at the steps (ii) and (iv) respectively of the generation algorithm mentioned in 4.3.2. Some of the generated sentences by a phrase structure grammar are shown in the following.

Often my friend always enjoy many good voyage to Kyoto

and sea.

You grow free.

He believe someone next.

He find the Mt. Fuji but the Lake Biwa dark.

To have them lead, the god think Japan bad.

Because we see, the bread seem blue but red or black and five.

The north look cold.

I know that they and she set of door.

I mark it to seem ten.

When he send us the Newsweek, Jack run.

These sentences are generated by the phrase structure grammar shown in Table 4.4. Although some sentences may be better explained by transformational rules, they are treated by a phrase structure grammar for the first experiment. The followings are the results of the application of transformational rules to the kernel sentences.

{You go
Do you go?

{They send they (obj)
Do they send they (obj)?

{They will have be (en) ask(ing) they(obj) often.
Will they have be(en) ask(ing) they(obj) often?

{Betty ask you of you.
Betty do not ask you of you.

{He must have be(en) go(ing) of the strong idea.
A false idea shall have be(en) after it.

Table 4.4 Expansion rules

SE → (LAR·S1·PRD (70)
LAR·SK·SN·PRD (30)

SK → CCO·SN·CMA (100)

S1 → ((AB1(4))·SN (80)
PP·CMA·SN (20)

SN → (NP/13/·PR (70)
NP/13/·PR·AND·PR/11/ (30)

PP → (PRP·NQ (50)
TOO·PR (30)
PRP·PR ing (20)

NP → (R1 (50)
R3 (35)
R2 (15)

NQ → (R1 (50)
R4 (35)
R2 (15)

NR → (R1 (70)
R2 (30)

R1 → (N1 (70)
N2 (30)

R2 → (RD (70)
RE (30)

R3 → (NO4 (70)
RF (30)

R4 → (NO5 (70)
RG (30)

N2 → (N1·AND·N1/11/ (70)
N1·CMA·N2/11/ (30)

RE → (RD·AND·RD/11/ (70)
RD·CMA·RE/11/ (30)

RF → (NO4·AND·NO4/11/ (70)
NO4·CMA·RF/11/ (30)

RG → (NO5·AND·NO5/11/ (70)
NO5·CMA·RG/11/ (30)

RD → ((BBB(5))·NM·(PP(5)) (65)
THE·NO3(PP(5)) (35)

N1 → NA·(PP(5)) (100)

```
NA --> ( NB          (40)
       | AA·NB       (30)
       \ BB·NB       (30)

NB --> ( NC          (80)
       \ AJ1·NC      (20)

NC --> ( ND          (80)
       \ AJ2/12/·NC  (20)

ND --> ( NE          (80)
       \ AJ3/12/·ND  (20)

NE --> ( NO1         (40)
       | NO1·NO1     (20)
       | NO2·NO1     (20)
       \ NO3·NO1     (20)

NM --> ( NG          (60)
       \ AJ2/12/·NM  (40)

NG --> ( NO2         (70)
       \ AJ3/12/·NG  (30)

PR --> ( (AB4(3))·BE·(PRP/15/·NQ/16/)(5)        (15)
       | (AB4(3))·BE·(TOO/15/·PR/16/)(5)        (15)
       | (AB4(3))·BE·(PRP/15/·PRing/16/)(5)     (10)
       | (AB4(3))·VP·(PRP/15/·NQ/16/)(5)        (15)
       | (AB4(3))·VT·(TOO/15/·PR/16/)(5)        (15)
       | (AB4(3))·VT·(PRP/15/·PRing/16/)(5)     (10)
       \ (AB4(3))·VTA·TAT·SN                    (20)

BE --> ( VI1         (30)
       | VI2·CM/14/   (40)
       \ VI3·CN/14/   (30)

VT --> ( VT1·NQ/14/          (20)
       | VT2·NO5·NR/14/      (40)
       | VT3·W1/14/          (20)
       | VT4·W2/14/          (20)
       \ VT5·W3/14/          (20)
```

W1 → NQ·NR/11/ (30)
NQ·CN/12/ (40)
NQ·VI1/31/ (30)

W2 → NQ·NR/11/ (30)
NQ·AJ2/12/ (30)
NQ·AJ3/12/ (20)
NQ·AJ1/12/ (20)

W3 → NQ·NQ/11/ (100)

CM → NP (65)
CN (35)

CN → AK (60)
AJ1·AND·AK (25)
AJ1 (15)

AK → AJ (75)
AJ·AND·AK/11/ (25)

AJ → AJ2 (70)
AJ3 (30)

AA → AAA (40)
BBB (30)
THE (30)

BB → BBB (50)
THE (50)

LAR : sentence top
CMA : comma
PRD : period
AAA : indefinite article
THE : definite article
BBB : pronoun (possessive case)
AND : coordinate conjunction
CCO : subordinate conjunction
AB1, AB4 : adverb
PRP : preposition
AJ1 : quantitiative adjective
AJ2, AJ3 : qualifying adjective
NO1 : common noun
NO2 : material noun
NO3 : proper noun
NO4 : pronoun (nominative case)
NO5 : " (objective case)
VI1 : complete intransitive verb
VI2 : incomplete intransitive verb
VI3 : special intransitive verb
VT1 : complete transitive verb
VT2 : " " (double objects)
VT3 : incomptete transitive verb
VT4, VT5, VTA : special transitive verb
TOO : to
TAT : that
PR ing : ing form

He must have be(en) go(ing) of the strong idea that shall have be(en) after it.

He may have be(en) sailing on the beautiful sea.
Sea be great.
He may have be(en) sail(ing) on the beautiful sea that be great.

At the present the morphophonemic rules are not programmed in the generation system, so that the above sentences are incorrect in the number, case, past participle form, present participle form etc. The characters in the brackets are inserted manually for the easy reading.

4.8 Considerations

Whether these generated sentences are admissible or not is very hard to judge. The difference between these sentences and those generated by V. H. Yngve may not be seen at a glance, but if we examine very closely, there are not such meaningless sentences as

Water is polished.

A black smokestack is proud and polished.

of Yngve in the sentences generated here. The experiment of Yngve was done by very few words such as 16 nouns, 7 verbs, 7 adjectives and so on while the experiment here is done by 500 words, so that the comparison of the quality of the generated sentences between these is meaningless. This comes from the fact that the relations among the semantic categories are very difficult and complex to set up, when the number of

categories and words increases considerably and when the meaningless sentences are to be prevented to appear.

A noun phrase which appears in a subject is in general simpler in structure than a noun phrase which appears in a predicate. However in some instances the structure of a subject is more complex than that of a predicate.

These structural unsymmetries are very difficult to realize in the generation. One way is to adjust the adoption ratio of a phrase explained in 4.7.1 properly. That is, when a prepositional phrase is adopted, the adoption ratio of another prepositional phrase is decreased which modifies a word in the prepositional phrase already adopted.

The results of this generation can directly be applied to the mechanical translation. For example the sentences like

I bought a car with four dollars.

I bought a car with four wheels.

are analyzed structually the same in every method. However if the semantic consistency is examined in every possible phrase just as it is examined in a phrase by the generation, the first sentence is known as having the structure,

(bought (a car)(with four dollars)),

while the second has the structure,

(bought ((a car)(with four wheels))).

From the procedure of sentence generation and the fact mentioned here, the boundary of the structure to be treated by a phrase structure grammar and by a transformational grammar might well be defined as follows.

If every phrase of a sentence is semantically consistent, the sentence can be explained by a phrase structure grammar, and if not, the sentence must be transformed by transformational rules to a set of kernel sentences which can be treated by a phrase structure grammar.

By this criterion compound and complex sentences can be correctly analyzed. For example,

People who apply for marriage licenses wearing shorts or pedal pushers will be denied licences.

can be considered as the composition from the following kernel sentences.

People apply for marriage licenses.

People wear shorts.

People wear pedal pushers

$\times$ will deny licenses to people.

A similar sentence,

Students who apply for university courses requiring assignments or laboratory experiments will be dedicated students.

can be considered as the composition of the following sentences.

Students apply for university courses.

University courses require assignments.

University courses require laboratory experiments.

Students will be students.

Students declicate themselves.

These decompositions of sentences to kernels require the minute semantic informations. However this kind of sentence decomposition suggests a wider range of computational linguistics,

such as question answering,[41] abstracting and so on.

4.9 Segmentation of Japanese Sentences

By the Japanese English translation one of the problems in the Japanese language is the segmentation of a Japanese sentence into word level. Japanese sentences are composed of Chinese characters and Kana letters. This is an important information in the segmentation of a Japanese sentence, although this is not enough. Another important key is Japanese auxiliary verbs and postpositions, which are rather few in number and which are never written in Chinese characters. In this section a segmentation experiment is illustrated, which was done on the Japanese sentences written all in Kana letters.

4.9.1 Connectivity of Parts of Speech

For the separation of each word in a Japanese sentence, connectivity of adjacent words are to be tested referring to the grammatical rules among parts of speech. Especially the finite forms of verbs, adjectives and auxiliary verbs restrict strongly the parts of speech which succeed them, so that these informations are to be utilized properly.

The followings are the list of parts of speech which precede another part of speech. For verbs, adjectives, and auxiliary verbs seven conjugation forms generally exist, which are negative, adverb, present, adjective, conditional, imperative, and future. Hereafter these are referred to as the first conjugation form, the second conjugation form etc..

(1) Ahead of a verb

(i) The second conjugation form of a verb.

(ii) " " " " of an adjective.

(iii) adverb.

(iv) conjunction.

(v) postposition (few are exceptions).

(2) Ahead of a noun

(i) All the parts of speech except shu-joshi (postposition of the end).

(ii) By the conjugative words the fourth conjugation form.

(3) Ahead of an adjective

(i) All the parts of speech except noun and shu-joshi.

(ii) By the conjugative words the fourth conjugation form.

(4) Ahead of a rentaishi and an adverb

(i) All the parts of speech.

(ii) By the conjugative words the second and the fourth conjugation forms.

(5) Ahead of a conjunction

(i) Noun.

(ii) By the conjugative words the second, third and the fourth conjugation forms.

(6) Ahead of an auxiliary verb

There are several kinds of connectivities. Connectivity table is prepared for each auxiliary verb.

(7) Ahead of an postposition

There are several kinds of connectivities. Connectivity table is prepared for each postposition.

(8) Ahead of a comma

(i) Noun, (ii) Adverb, (iii) Conjunction,

(iv) postposition,

(v) The second conjugation form.

(9) Ahead of a period

(i) postposition of the end

(ii) The third and the sixth conjugation form.

(10) Parts of speech which do not come to the top of a sentence.

(i) Auxiliary verb, (ii) Postposition

4.9.2 Algorithm of the Segmentation of a Sentence into Words

A sentence is supposed to be composed of n Kana letters ($i = 1, 2, \ldots, n$), such as

$$S = a_1 \cdot a_2 \cdot \cdots \cdot a_n .$$

When a word is denoted by wi, then the object is to transform the above sentence into

$$S' = w_1 \cdot w_2 \cdot \cdots \cdot w_m \qquad m \leq n$$

where

$$w_1 = a_1 \cdot \cdots \cdot a_i , \quad w_2 = a_{i+1} \cdot \cdots a_j , \cdots , \quad w_m = a_{k+1} \cdots a_n .$$

Also the following conditions are necessary.

(i) w_1, w_2, $\cdots$, w_m must be proper Japanese words (which are expected to be found in a dictionary or to be recognized by certain algorithm).

(ii) w_1 and w_2, w_2 and w_3, ..., w_{m-1} and w_m, each must have correct connection relation according to the Japanese grammar (in this experiment the above 10

restrictions are to be satisfied).

The transformation from S to S' is not unique even if the above conditions are satisfied. Therefore all the possible segmentations are to be produced for the later analysis by semantic informations.

The operations to be performed are,

(1) to devide a sentence S into a series of words w_1, w_2, ..., w_m,

(2) to know the part of speech, and conjugation form of each word,

(3) to test whether the connection to the adjacent words are permitted or not.

To do these three operations efficiently the steps shown in the flow chart of Fig.4.7 are taken. The program determines each word from the top of a sentence successively by testing the connectivity to the left element until finally the end of a sentence is reached. If the connectivity test is failed at a certain point the longer word is taken out. When one segmentation is completed it is memorized in other place. Then its last word is discarded and the second word from the last is made one character longer and the same operation is resumed to search for the other segmentation. The end of the program is when there remains no other segmentation at all. Actually the word length is limited within five characters, so that if a string of five characters does not form a word, it is abondoned and the process goes back to one word backword. This path is indicated by dotted lines in the flow chart.

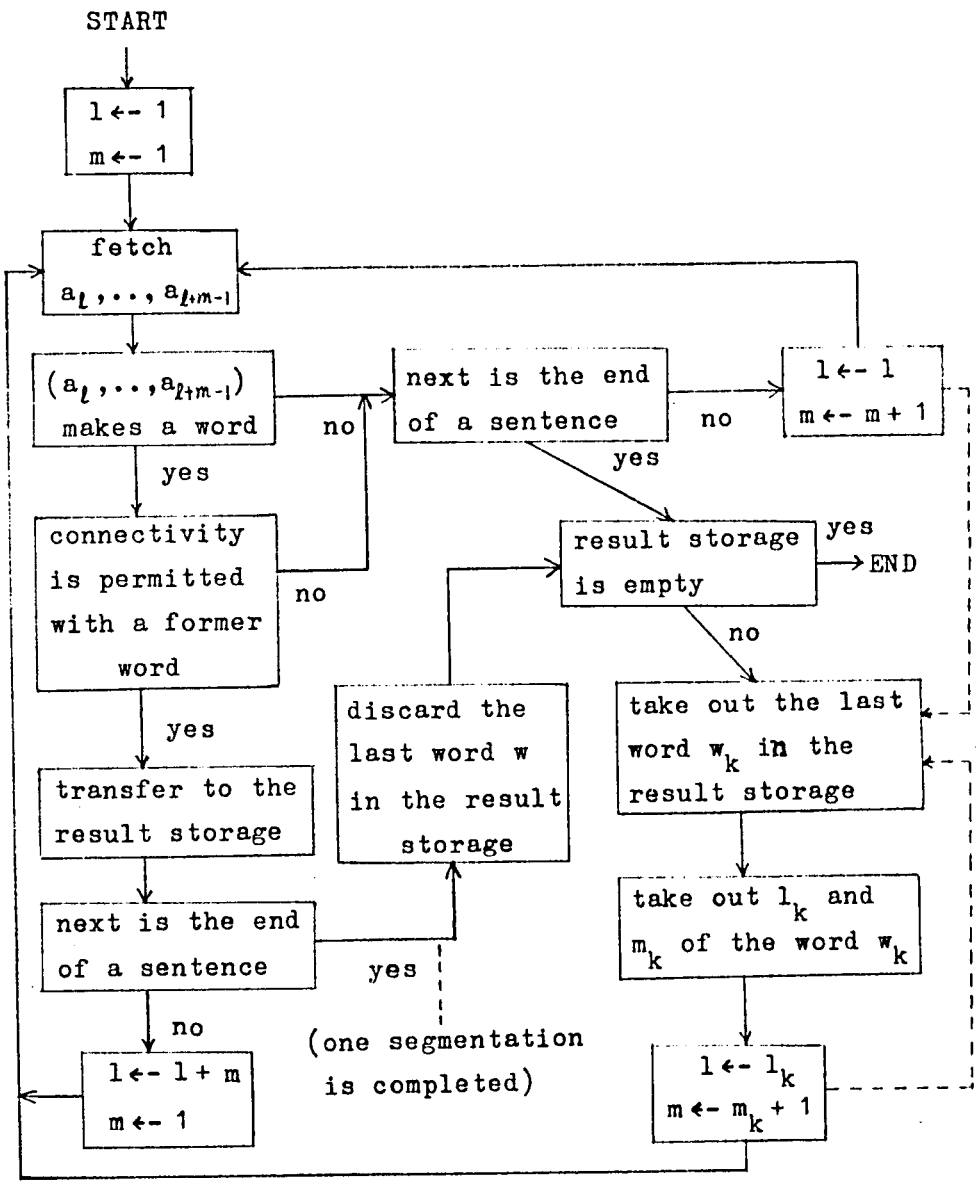


Fig 4.7 Flow chart of the segmentation of a sentence.

4.9.3 Dictionary

(a) Presentation of Kana letters in computer number system. The computer the experiment was done is KDC-I, which is a decimal machine with 11 degits word length. The restriction that a word is within five characters comes from this. However it is not so inconvenient because longer words can be treated as a compound word of two or three.

Each Kana character is coded by 2 digits. The five vowels A, I, U, E, and O are coded as 01, 02, 03, 04, 05, while KA, SA, TA, NA, HA, MA, YA, RA, and WA are coded as 11, 21, 31, 41, 51, 61, 71, 81, and 91 respectively.

Voiced consonants, such as GA, ZA, DA, etc. are coded as the code of the corresponding unvoiced consonant plus five.

Consonants PA, PI, PU, PE, and PO are coded as 66, 67, 68, 69, and 70 respectively.

Special characters N, ゛, ゜ are coded as 97, 98, 99.

Contracted sounds were not treated, so that kya, sha, etc. are to be written as kiya, siya etc.

Also assimilated sounds were not treated, so that itta is to be written as ituta. Euphonical changes in Japanese are not so frequent that they are treated by the program.

(b) Dictionary

In a dictionary the following items are stored.

(1) Japanese word 5 characters in 10 digits.

(2) Degree of multiplicity 1 digit.

(3) part of speech 2 digits

(4) Conjugation form 1 digit

(5) Distinction of voiced and unvoiced sounds 1 digit

(6) Address of the connection table for postposition and auxiliary verb 4 digits

All the conjugation forms of the verbs of irregular conjugation, i.e. SURU and KURU, and all the auxiliary verbs are listed in the dictionary. For the verbs such as KAKU whose 7 conjugation forms are KAKA(1111), KAKI(1112), KAKU(1113), KAKU(1113), KAKE(1114), KAKE(1114), and KAKO(1115), only the constant part, in this case (111), is stored as entry word.

Degree of multiplicity means that the same Kana word belongs to several different words, such as SIRO is castle and white. There are provided as many entries as the degree of multiplicity. So up to nine multiplicities are possible. This treatment of multiplicity is abbreviated in the flow chart of Fig. 4.7. The dictionary size was 200 words by the experiment, and the maximum degree of multiplicity was three.

4.9.4 Semantic Components

Usually there appears plurality of segmentations for a sentence by the method mentioned above. A key for the determination of a proper segmentation among the plurality lies in the semantics. The consistency of a sentence is very hard to determine especially when the semantics must be considered. One trial is presented in the former sections of this chapter. Here another trial is shown which is far more simpler than the former, and can be used very conveniently for the other fields

such as abstracting, indexing and so on. Also the data preparation is easier by this method than by the former one.

For the determination of a proper sentence among several possible segmentations, the first thing is to see the mutual relations of the nouns included in the sentence. Nouns are classified very roughly into the following five categories.

(1) Nouns of abstruct relations.

In this category is included time, year, season, direction, space, dimensions, movement etc.

(2) Subject of human activity.

In this category is included religion, politics, community, occupation, family, etc.

(3) Activity of human being.

In this category is included spiritual action, phycal action, physiology etc.

(4) Products and tools of human action.

In this category is included meals, clothes, home, tools, buildings etc.

(5) Natural phenomena.

Natural phenomena, animals, plants, celestial sphere, etc. are included in this category.

Among these five categories semantic distances are introduced in the following way. These five categories are regarded as the five tops of a pentagon, and each side of it represents the distance. Here three kinds of distances are set up.

(1) Between the nouns belonging to the same category,

the distance is d_1.

(2) Between the nouns belonging to the categories of the adjacent tops, the distance is d_2.

(3) Between the nouns belonging to the categories of the tops which hold a third top in between, the distance is d_3.

Now when there are n nouns in a sentence, the combinations of 2 of these are ${}_nC_2$. Among these combinations it is supposed that n_1 pairs belong to the distance d_1, n_2 pairs belong to the distance d_2, and n_3 pairs belong to d_3 ($n_1 + n_2 + n_3 = {}_nC_2$). From these

$$D = \frac{n_1 d_1 + n_2 d_2 + n_3 d_3}{{}_nC_2} = \frac{n_1 d_1 + n_2 d_2 + n_3 d_3}{n_1 + n_2 + n_3}$$

is calculated. When there are many nouns which belong to the same category or to the nearest categories, D becomes small. In this experiment the segmentation whose D is the smallest is regarded as the best one, although this has no definite reasoning. In other words by calculating D, it is tested how far the distinction can be done among different possible structures.

Few examples of the segmentation experiment are shown in the following, where $d_1 = 1$, $d_2 = 2$, and $d_3 = 3$.

(1)
INAKA NO SIZEN HA TASIKANI UTUKUSII. (D = 2.00)
(The nature of the country is surely beautiful.)
INAKA NO SIZEN HA TASIKANI UTUKUSII. (D = 2.33)
(The natural leaf of the country is surely beautiful.)
HITO HA TUSIN SURU DOBUTU DEARU. (D = 2.67)

(2) (Man is an animal who communicates.)
HI TOHA TUSIN SURU DOBUTU DEARU. (D = 2.33)
(Fire is an animal which communicates.)

(3) SOUIU UTUKUSISA MO NARERU TO UTUKUSISA O KANZI NAKU NARU DARO U TO IWU HITO MO ARU GA SOU TOMO KAGIRA NAI. (D = 1.53)
(Some people say that such kind of beauty may not be appreciated when it becomes very familiar for us, but it is not always true.)

(4) SORA NO IRO DEMO KI NO HA NO IRO DEMO, TOKAI DE MIRU MONO TO MARUDE TIGATU TE IRU. (D = 1.86)
(Everything looks completely different here from those in the town, such as the color of the sky and the leaves of the trees.)

4.9.5 Consideration

For the Japanese English translation by mechanical means, the segmentation of a Japanese sentence into words is essential, whether this is done by hand as preprocessing or by machine. When a sentence is given in ordinary style of Chinese characters and Kana letters, the segmentation may be a bit easier than by the sentences of all Kana letters. However in this case another problem appears, i.e. the coding of Chinese characters.

This experiment is not satisfactory in many respects. Dictionary size is very small, the semantic categories introduced are not enough, and there must be more fundamental data

for the determination of semantic distances. Also there arises a doubt whether this simple semantic treatment without any syntactic informations is enough or not. For the successful segmentation syntactic structure will be necessary, whereas for the determination of syntactic structure the sentence must have been segmented into words first of all. This is apparently contradictory. Therefore a method is to know all the possible structures for all the possible segmentations. Here the most important problem is to find a method to determine the best among many possible candidates. It is not still known for us but it may be the over-all linguistic desctiption including syntax and semantics.(42)

CHAPTER 5

CONCLUSION

Throughout the PART I of this thesis the fundamental attitude was to know the complete structure of a language, either is it a formal language or a natural language. By formal languages the structural description is definite, so that if the language has no ambiguity the syntactic analysis is unique, and the analysis algorithm is obtained as explained in Chapter 2. However there arise sometimes the cases where the structural description is not adequate. For example ALGOL 60 defined by Backus normal form has definite construction rules, but has no definite interpretation rules. This latter is described by a natural language quite conventionally, so there remain many special cases where no definition is given for interpretation. One of these is pointed out at the end of 2.4.

There are other kinds of problems of semantic ambiguities. For example the same variable can be used independently for any meaning in different blocks. This fact is very difficult to express by some formal representations.

Likewise there are many interesting problems to be solved concerning to Chapter 2. Among them the ambiguity problem of operator language must be solved first of all. Semantic problems will become very important in formal language, especially in programming language in the coming few years. The application of the analysis method developed in Chapter 2 on ALGOL

60 may be very interesting, in which case ALGOL 60 must be slightly deformed to match the requirement of operator language. Here the semantics of ALGOL 60 must be defined in formal representation especially for the handling of declarations. How the formalization of semantics must be done and can be done is most important problem for formal languages as well as for natural languages.

Chapter 3 is a study of structural analysis of general context-free phrase structure languages. An important point is that languages are assumed to have structural ambiguity. The analysis algorithm must be necessarily the form which examines all the possible pathes. A method will be the study of minimizing the total steps of analysis algorithm referring to some special symbols or else, because for the search of all possibilities a great amount of time is necessary and in such cases usually some heuristic methods are introduced. Here however the aim was the analysis algorithm of a substring from the top to any point of a given string. And the algorithm is to find out not only sentence structures but also any phrase structures, such as noun phrase, predicative phrase and so on.

When an analysis algorithm can find all the possible structures the grammar construction for a natural language becomes very easy. On the contrary if an algorithm is restricted to find only a structure, the grammar construction becomes too difficult to do, so that the former method is imperative.

The algorithm explained in Chapter 3 is very powerful for

the analysis of general formal languages and natural languages. This is however restricted to the context-free phrase structure languages. The study of the properties and analysis algorithms of context-restricted phrase structure language must be done extensively. For example, transformational grammars are not context-free phrase structure grammars, but are included in general context-restricted phrase structure grammars. However this grammar is too general for some useful results to be produced.

In Chapter 4 semantic treatment is studied as an important key in the structural analysis of natural languages. From the standpoint of computational liguistics and mechanical translation studies, semantics as has been developed in the past is of no use. It is because semantics must have very close relation with syntax. For the study along this line synthetic method is adopted. This clearly presents many specialities of natural languages which we do not realize at all. The proposed generation process might be different from that of human being, but it clearly showed what kinds of semantic informations are necessary for the processing of natural languages. The position of transformational rules is clarified in accordance with semantics. However transformational rules have more closer connection with semantics than phrase structure grammars, so that how to combine transformational rules with semantics is an important problem to be solved very urgently, as well as the formalization of the rules in accordance with phrase structure grammar.

Another trial on semantics is shown in the study of the segmentation of Japanese sentences. The aim was to see how the difference between sentences can be recognized by such small semantic informations. This kind of approach is very useful for the studies which must process a great amount of documents, such as abstracting, indexing etc.

The digital computer KDC-I is too small for the processing of languages. High speed, large memory capacity of rapid access, large back-up memory, card base etc. are essential. Also there must be a convenient programming language particularly suited for language processing.

BIBLIOGRAPHY

(1) Reference Manual, 709/7090 FORTRAN Programming System, IBM Form No. C28-6054-2.

(2) P. Naur, et al,, "Report on the algorithmic language ALGOL 60," Comm. ACM, Vol.3, pp.299-314, May, 1960.

(3) P. Naur, et al., "Revised report on the algorithmic language ALGOL 60," Comm. ACM, Vol.6, pp.1-7, Jan., 1963.

(4) J.W. Backus, "The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference," Proc. Internat'l Conf. Information Processing, UNESCO, Paris, pp.125-132, June, 1959.

(5) N. Chomsky, "On Certain Formal Properties of Grammars," Inf. and Control, Vol.2, pp.137-167, June, 1959.

(6) M. Gross, "On the Equivalence of Models of Language Used in the Field of Mechanical Translation and Information Retrieval," Mimeograph, April, 1962.

(7) R.J. Evey, "The Theory and Applications of Pushdown Store Machines," Doctoral thesis, Harvard University, 1963.

(8) R.A. Brooker and D. Morris, "A general translation program for phrase-structure languages," J. ACM, Vol.9, pp.1-10, Jan., 1962.

(9) E.T. Irons, "A syntax directed compiler for ALGOL 60," Comm. ACM, Vol.4, pp.51-55, Jan. 1961.

(10) R.W. Floyd, "Syntactic analysis and operator precedence," J. ACM, Vol.10, pp.316-333, July, 1963.

(11) W.N. Locke & A.D. Booth (ed.), "Machine Translation of Languages," John Wiley & Sons, 1955.

(12) J.J. Katz & J.A. Fodor, "The Structure of a Semantic Theory," Language, Vol.39, No.2, 1963, pp.170-215.

(13) M. Nagao, "An Approach to the General Theory of Natural Languages," Preprints for Seminar on Mechanical Translation, U.S.-Japan Committee on Scientific Cooperation,

Pannel II, April, 1964.

(14) T. Sakai & M. Nagao, "Sentence Generation by Semantic Concordance," Proc. 1965 Internat'l Conf. on Computational Linguistics, May, 1965.

(15) V.H. Yngve, "Random Generation of English Sentences," 1961 International Conference on Machine Translation of Languages and Applied Language Analysis, London, 1961.

(16) M. Nagao, "The structural analysis of phrase structure grammar," J. Information Processing Society of Japan, Vol.4, No.4, July, 1963.

(17) M. Nagao, "Syntactic Analysis of Phrase Structure Languages," Information Processing in Japan, Vol.4, 1964.

(18) P.Z. Ingerman, "A translation technique for languages whose syntax is expressible in extended Backus normal form," Symbolic Languages in Data Processing, Gordon and Breach, 1962.

(19) N. Chomsky and M.P. Schutzenberger, "The Algebraic Theory of Context-Free Languages," Computer Programming and Formal Systems, Ed. by P. Braffort et al., North Holland, 1963, pp. 118-161.

(20) K.E. Iverson, "A Programming Language," John Wiley & Sons, 1962.

(21) R.W. Floyd, "The Syntax of Programming Languages--A Survey," Comm. ACM, Vol.7, pp.346-353, August, 1964.

(22) K. Samelson and F.L. Bauer, "Sequential formula translation," Comm. ACM, Vol.3, No.2, 1960, pp.76-83.

(23) M. Nagao, "About the Language Structure of ALGOL 60," Record of the 1963 Joint Convention of the Kansai District of IECEJ* and others. S11-2, 1963.

(24) M. Nagao, "Structural Analysis of General Context-Free Phrase Structure Languages," Record of the 1963 Convention of the Information Processing Society of Japan, 1963.

(25) S. Kuno and A.G. Oettinger, "Multiple-Path Syntactic Analyzer," Proc. IFIP 62, North Holland, 1963.

(26) C.F. Hockett, "Grammar for the Hearer," Structure of Language and Its Mathematical Aspects (R. Jakobson ed.), Proc. Symposia in Applied Mathematics, Vol.12, American Mathematical Society, 1961.

(27) A.G. Oettinger, "Automatic Language Translation," Harvard University Press, 1960.

(28) N. Chomsky, "Syntactic Structures," Mouton & Co. The Hague, 1957.

(29) M.P. Schutzenberger, "On context-free languages and push-down automata," Inf. and Control, Vol.6, pp.246-264, Sept., 1963.

(30) Y. Bar-Hillel, "The Present Status of Automatic Translation of Languages," Advances in Computers I, 1960, pp.92-163, Academic Press, 1960.

(31) P.L. Garvin, ed. "Natural Language and the Computer," McGraw Hill, 1963.

(32) P. Postal, "Constituent Structure," Mouton & Co. The Hague, 1964.

(33) D.G. Hays & T.W. Ziehe, "Studies in Machine Translation -10: Russian Sentence-Structure Determination," RM-2538, The RAND Corporation, April, 1960.

(34) I. Rhodes, "A New Approach to the Mechanical Translation of Russian," Report No. 6295, NBS, Washington, D.C., 1959.

(35) C.K. Ogden, I.A. Richards, "The Meaning of Meaning," 4th ed., London, 1936.

(36) S. Ullmann, "The Principle of Semantics," Basil Blackwell and Mott Ltd., 1951.

(37) C.E. Osgood & G.J. Suci, P.H. Tannenbaum, "The Measurement of Meaning," Univ. of Illinois Press, 1957.

(38) "Roget's Thesaurus," Penguin Reference Books R7, Penguin Books.

(39) A.D. Booth & A. Colin, "On the Efficiency of a New Method of Dictionary Construction," Inf. and Control, Vol.3, pp.327-334, 1960.

(40) O. Hayashi, "Bunrui Goi Hyo(Table of Japanese Word Classification)" Shuei-sha, 1965.

(41) R.F. Simmons, "Answering English Questions by Computers: A Survey," Comm. ACM, Vol.8, No.1, Jan. 1965.

(42) J.J. Katz, P.M. Postal, "An Integrated Theory of Linguistic Descriptions," The M.I.T. Press, 1964.

*IECEL : The Institute of Electrical Communication Engineers of Japan.

PART II

DESIGN OF ASYNCHRONOUS CHARACTER RECOGNITION MACHINES

CHAPTER 1

INTRODUCTION

1.1 Character Recognition

In discussing pattern recognition, there always arises a problem; what is meant by "pattern"? If pattern is restricted to some characters or symbols, the problem becomes somewhat clear. Actual patterns are the realization of abstruct symbols to which they are to be reduced. This reduction process to the abstruct symbols may be said as a problem of character recognition. There are so many patterns which are to be regarded as a same symbol, that the reduction is usually done by abstructing proper invariant parameters which commonly exist for a group of patterns to be reduced to a symbol. These parameters are to be sufficient in the sense that they can separate distinctly the patterns of different symbols. But this is a different problem from whether the combination of these parameters can reproduce the original patterns or not. If it can, the parameters are complete, and the recognition scheme does not depend on the number and kinds of characters included other than itself. If it can not, the recognition scheme largely depends on them, which is the usual case in actual character recognition.

However the parameters to be extracted should be invariant against shape, size, position, thickness, inclination, extraneous ink, voids and so on. There have been many investigations for the excellent parameters satisfying these condi-

tions, but still there is no decisive method for that.[1][2] On the other hand there are several studies for print qualities, scanning methods etc., which make the extracted parameters reliable.[3]

In character recognition there are many factors to be considered such as follows.

(1) The kind and number of characters to be read.

10 numerals are the characters most widely studied. Next is the capital English alphabet. For the Japanese, KATAKANA characters are important objects. HIRAKANA and Chinese characters are beyond our ability still now. Whether these characters are printed, typed or hand-written[4] is another important factor for the recognition. Most of the investigations are for the printed or typed characters of the same size, although few trials are there for the hand-written characters.

For the commercial use where high speed and high correct reading rate are required, some stylized character fonts are invented such as OCR-A[5] and CMC-7[6] etc.. Also there is magnetic ink printing as well as ordinary ink printing.

(2) Scanning method.

There are two distinct attitude for the scanning. One is to input the pattern images as accurately as possible, in which flying spot scanner, video scanner, etc. are classified. The other is to process the pattern images to a certain degree at the input stage, in which

the slit method, fixed point sampling, etc. are classified.

(3) The positioning of a character and the paper handling.

Patterns to be read are to be brought in to the region where scanner can accept. This paper handling is an important factor in character recognition machine. However there is a limitation to the preciseness of the mechanism. Therefore it is hoped that an accepted signal pattern is to be shifted in suitable position before the analysis procedure.

(4) Preprocessing of input patterns against noise, shape, size, boldness, inclinations and so on.

This is closely related to the recognition principle. However this process is often performed independent of the recognition principle for the elimination of noise, normalization of line width etc.[7]

(5) Recognition principle.

There have been so many recognition principle for the pattern recognition that here everything can not be listed.[1][2][3] One way is to construct a special purpose machine and another is to utilize digital computer for the processing of digitized patterns. Another classification is whether the process includes the learning process or not.[8][9]

In any character recognition method the usefulness of the system should be judged by the reading speed, accuracy of correct reading rate, and cost.

The character reader constructed by the author has the following excellent points, and the system is believed to be highly suitable for the commercial use.[10]

(1) The machine can read multifonts of 10 numerals, capital English alphabet, and some other symbols.

(2) Ordinary typed characters are read optically.

(3) The machine can read a whole page of characters, scanning many lines continuously in very high speed, as well as a long tape where characters are typed in a line.

(4) The recognition principle is essentially asynchronous, so that the variations of scanning speed have no effect for the recognition.

(5) The recognition circuits are combinatorial logical circuits and asynchronous sequential logical circuits, both of which are very simple, so that the construction expenses are very low.

(6) The correct reading rate is very high. The error reading rate is less than 0.2% and the rejection rate is less than 0.1%.

1.2 Outline of the PART II

The part two of this thesis is an experimental study of character recognition. The emphasis is on the construction of a high speed high accuracy character recognition machine which is worth for practical usage.

Chapter 2 is devoted to the complete presentation of the character recognition machine constructed by the author. The

asynchronous recognition principle is first introduced, in which are explained the basic recognition principle, some modifications of the principle, inhibition logics for the discrimination of similar patterns, the handling of special characters and so on. Then the input device and some other auxiliary parts of the recognition machine are illustrated. Finally the recognition results of the machine are presented and the performance of the machine and the factors affecting the recognition results are thoroughly discussed from every respect.

Chapter 3 is the design of some other character recognition machines which are also asynchronous and may be thought as the variations of the machine presented in Chapter 2. The first one is the separate recognition of upper and lower subpatterns by the same principle.[11][12] This aimed at the decrease of the number of characteristic feature logics and also of the recognition circuits, but was not so successful. The second one provides two input parts of many channels vertically aligned in parallel. This input device is for the detection of the position of each character and is very effective for the special characters treated in Chapter 2. The third one[13] has the two dimensional input part which eliminates the memory devices required for the other recognition methods. These character recognition machines are not actually constructed, but the functionings have been simulated successfully by a digital computer.

Chapter 4 explaines a computer programming system which

is particularly suited for the simulation of logical circuits. (14)(15) This system was developed by the author for the simulation of several pattern recognition machine presented in Chapters 2 and 3. This system is new in the point that asynchronous circuits can be accepted as well as synchronous one.

CHAPTER 2

DESIGN OF AN ASYNCHRONOUS CHARACTER RECOGNITION MACHINE

2.1 Outline of the Machine

2.1.1 Design Principle

By the pattern recognition machines so far developed, the recognition criteria are so severe that for the patterns with noise or with slight modifications the correct reading rate decreases considerably. Therefore there are very strict limitations on the size, position, print quality, etc. of the characters, and the high precision is demanded for the mechanism of page feeder and scanner.

To save these deficiencies of the previous pattern recognition machines, the following points are especially taken into consideration to the design of this pattern recognition machine.

1. The real time reading of characters, or the very high speed reading.
2. The continuous reading of many lines of characters on a page.
3. The reading machine which should be very compact and low in cost.
4. Optical reading which does not require the special printing.
5. The characters to be read are not only numerals but also English alphabet.
6. The reading machine which can accept several kinds of fonts.

7. The high accuracy of reading rate which is enough for the practical machine.

There are two main trends for the research study of pattern recognition. One is by using the digital computer, in which case any conceivable operations can be done on the digitized input pattern.(3) The other is to construct a practical machine, in which case the recognition speed should be very high. Most of the researches of character recognition so far have been to recognize characters which are supposed to be positioned appropriately to the input part, and in which case the continuing of reading another characters was out of their studies. However if a character recognition machine aims at a practical use the high speed feeding of many characters to the input part is essential. An example is to read the whole page where there are many lines of characters.

We do not want to go into the comparison of superiority between MICR (Magnetic Ink Character Reader) and OCR (Optical Character Reader) methods, but it may be enough to say that the printing of MICR needs special and precise printing devices.

The cost versus recognition speed and accuracy is another important factor by the pattern recognition. It may be well known that the recognition accuracy can be brought up very high, if we adopt a very compricated method of recognition. So the practical machine is always the compromise between accuracy and cost.

When an input pattern is digitized, for example, 20 times 20, the information which can be expressed by this mesh is

2^{400}. However the patterns we want to recognize are perhaps less than one hundred, so that the problem is how to diminish efficiently the very great amount of information received to less than one hundred. Moreover digitization will need complicated devices and take much time for the sampling and also need a very large memory storages. Considering these disadvantages we have adopted a different scanning method. The scanning is done by a number of input channels which are aligned vertically and the input patterns are to be shifted horizontally. In this case the received signals are essentially asynchronous ones so that the asynchronous recognition logic circuits are adopted.

2.1.2 The Block Diagram of the Machine

The schematic block diagram is shown in Fig.2.1. The photo electric conversion is done by 22 photo-transistors, which are aligned vertically and the patterns come in horizontally. This situation is schematically shown in Fig.2.2. Theheight of a pattern is adjusted to that of 16 channels. Another 6 channels are provided for the mechanical deviation of the shifting up the paper for many line spacings. The base line detection circuit i.e. the line positioning circuit determines the position of a line, i.e. the proper 16 channels out of 22.

The scanning of a whole page is done as shown in Fig.2.3. First the scanning is done from the right end to the left, and then the same position is scanned from left to right. At the

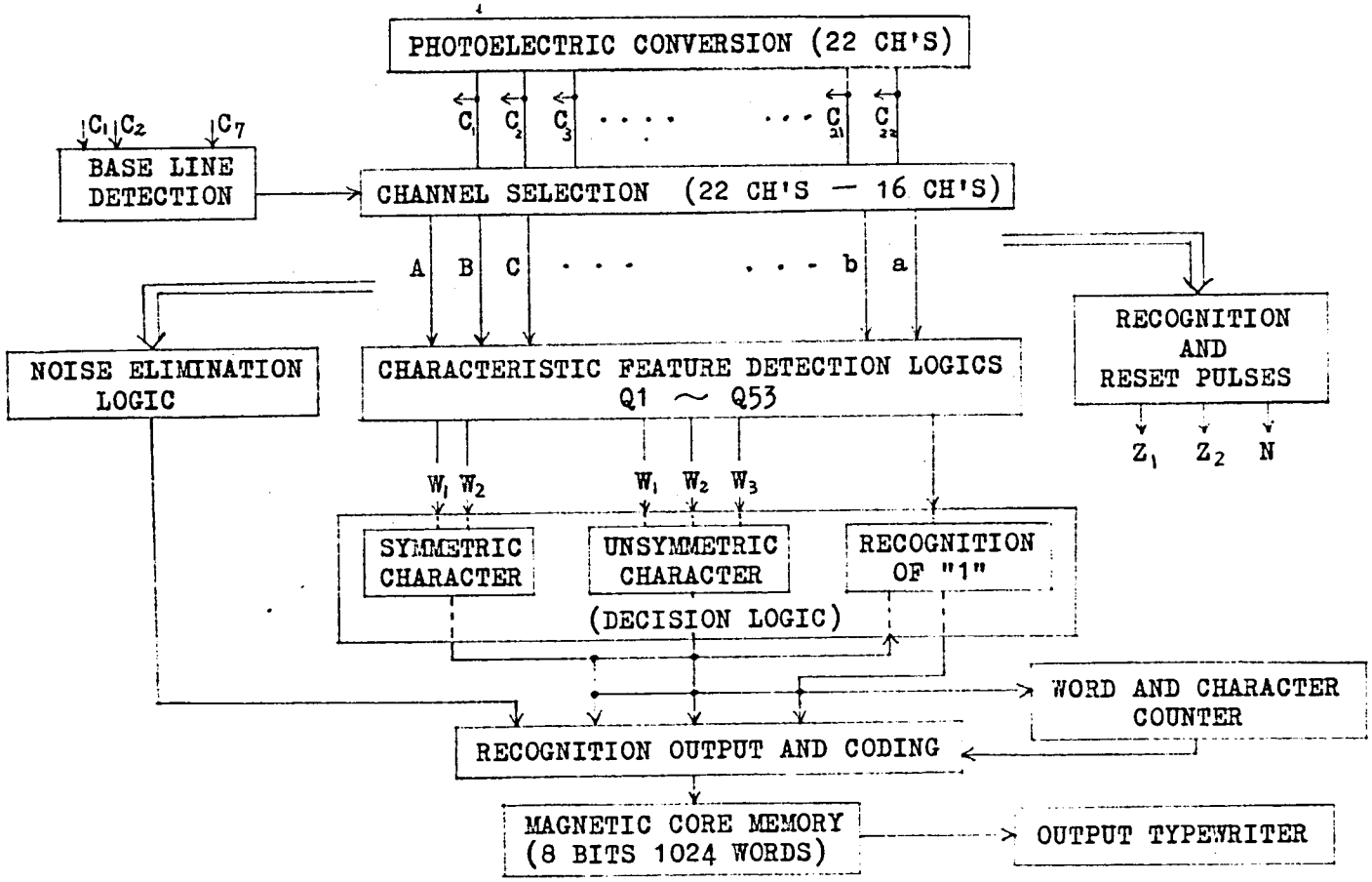


Fig. 2.1 Schematic block diagram of character recognition machine.

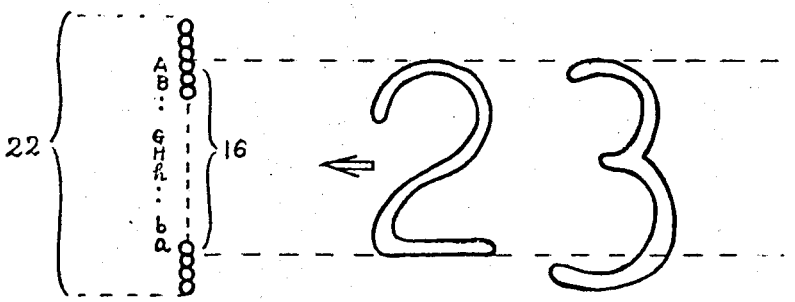


Fig. 2.2 Photo transistor input part and the movement of characters.

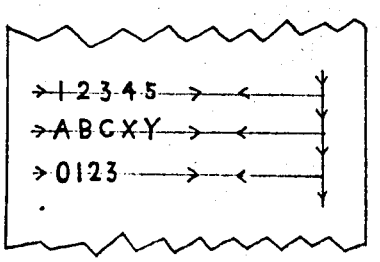


Fig. 2.3 Method of scanning a whole page.

end of this scan the paper is shifted one line spacing up and the same operation is done. During the first right to left scan the base line detection circuit operates and at the end of this scan the top channel of a line is detected. The channel selection logics connect the 16 channels, named A, B, C, ...G, H, h, g, ...b, a, from top to bottom, to the proper 16 channels where there is a line of characters, by the signal from the base line detection circuit.

The characteristic feature detection logics are the logical functions which are established among the 16 channels A, B,..., G, H, h, g,.....b, a, and which detect the features of vertical sections of characters best.

The recognition logics are provided for each character and work by the sequential appearance of signals from three characteristic feature logics for each input characters. The signals from the characteristic feature logics are not synchronous ones, but very irregular, so that only the sequence of signal appearance of characteristic feature logics is detected by asynchronous sequential logic circuit.

The recognition output is coded in 6 bits code and transferred to the core memory. The recognition result is printed out from the core memory and we can see how the recognition has been done. The recognition logic for character "1" is different from the others. The noise elimination logic works for extraneous ink and suppresses the error recognition output. The word and character counters are provided for the read-out of only the specified positions of a document by the counters.

Several control signal circuits govern the timing relations between the scanner and the electronic circuits, provide the recognition interval, and reset all the circuits to the initial states. The machine also provides the display for the monitoring of input patterns and the alarm signal circuit which works when a printed line largely deviates from the planned region for the input. Fig.2.4 is the overall view of the character recognition machine.

2.1.3 Character Fonts

The structure of a character recognition machine largely depends on the number and the nature of character fonts which the machine wants to read.

Concerning to the number, 10 numerals may be the first aim of the recognition. Next 10 numerals plus some special symbols. For the ordinary purposes of the business field this repertory may be enough. Therefore the character recognition machines of several years ago were of this kind. The most famous one is MICR (Magnetic Ink Character Recognition) adopted for check handling by the American Bankers Association (ABA).

However the demands for the recognition of more characters, as far as 10 numerals plus capital English alphabet naturally have arisen from the business field and also from the research projects of machine translation of natural languages especially from Russian to English in the U.S.A. Nowadays the kinds of characters normally understood to be included in the recognition machine are 10 numerals, capital English alphabet, and a few

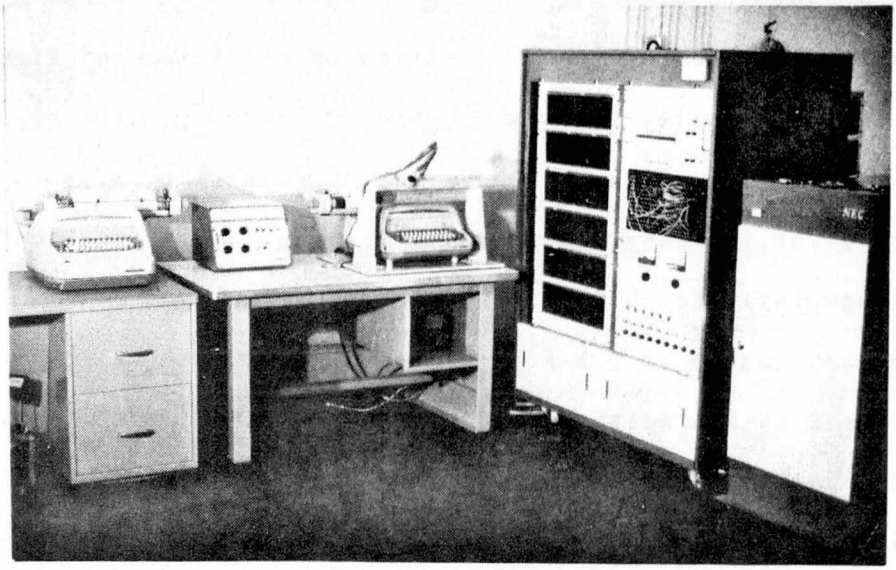
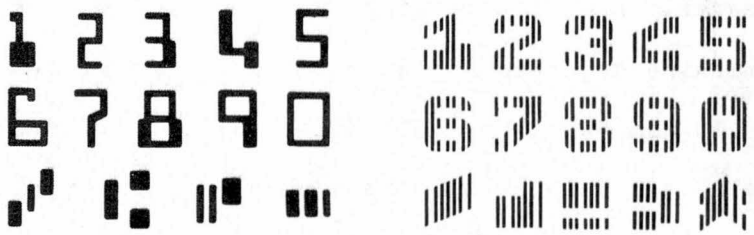


Fig. 2.4 Overall view of the character recognition machine. From left to right; output printer, control panel, input scanner, recognition part, magnetic core memory.



(a) (b)

Fig. 2.5 Character fonts for MICR. (a) E13B, (b) CMC7.

special symbols.

As for the nature of character fonts, there are too many standpoints to be settled in a direction. The first aim, however, may be said the recognition of characters of single font. Especially for the commercial use, where the high speed, high accuracy reading is essential, special type fonts are invented for the easy and accurate reading. The font recommended especially for MICR is an example, which is shown in Fig.2.5. Another example is OCR-A for the optical character recognition proposed by ASA(American Standard Association) to International Organization for Standardization (ISO). This is shown in Fig. 2.6.

There are many other kinds of fonts especially designed for the ease of the reading. These fonts are usually called stylized fonts, and are distinguished from the ordinary fonts. There are many experimental machines which can accept several kinds of fonts and arbitrary size of characters. Moreover there are several experimental machines which recognize hand written numerals. These machines are not practical yet, but it will prevail in the near future.

The machine designed and constructed here aims at the reading of 10 numerals, capital English alphabet and some special symbols. It also aims at the reading of several kinds of fonts as far as possible. The characters which the machine primarily intends to read are,

numerals : NEAC TYPER font, OCR-A (ISO); 0--9

alphabet : NEAC TYPER font; (A,B,C,K,N,P,R,S,U,V,X,Y)

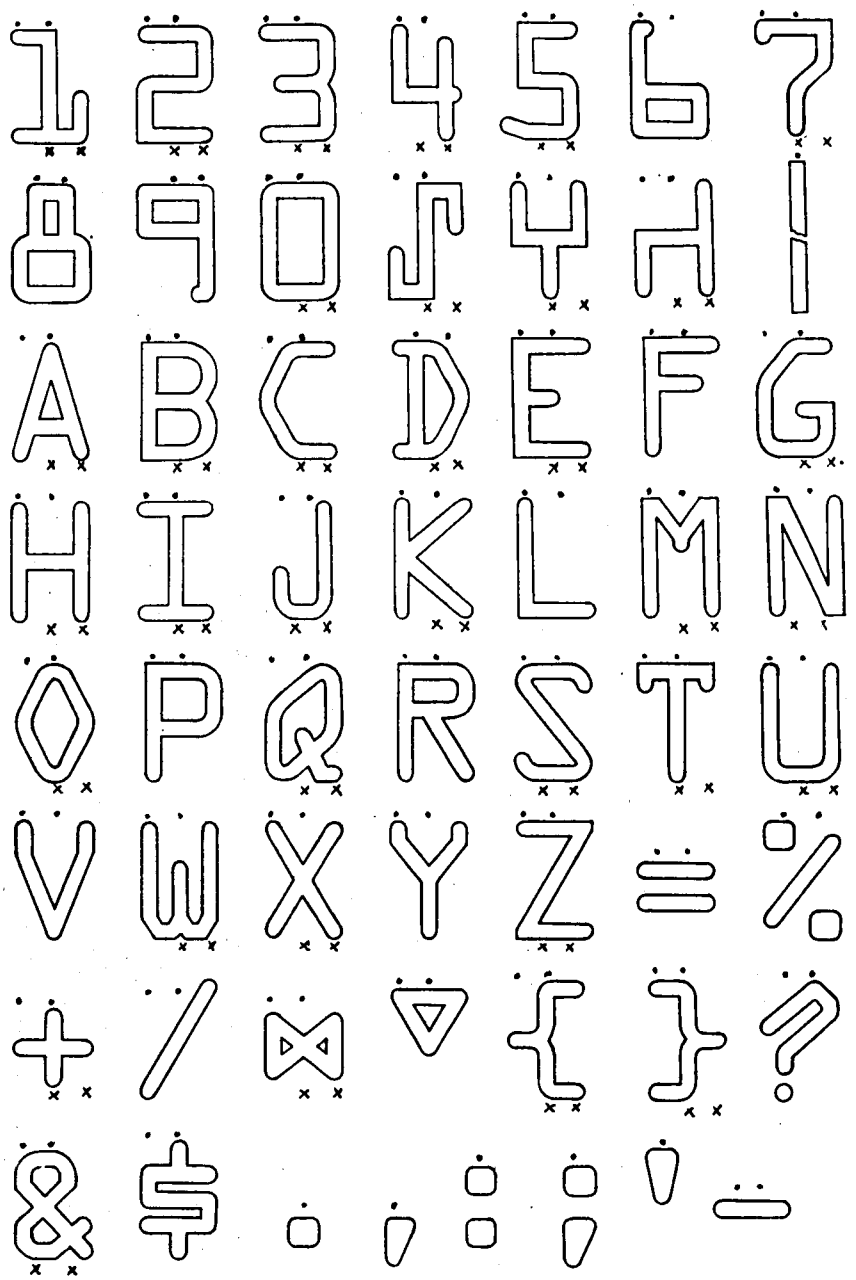


Fig. 2.6 OCR-A type font proposed by ASA. • and × are for the explanation in 3.2.2.

Fig. 2.7 NEAC TYPER font.

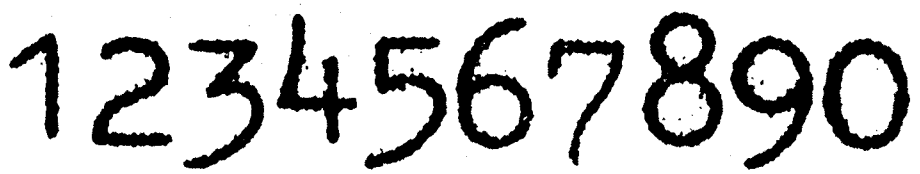


Fig. 2.8 Different character heights and misalignment of character positions.

special symbol : < , > ; ⌐ , ⊤, ⊓, (OCR-A)

From the restriction of the budget all the capital English alphabet could not be included in the machine. The characters of NEAC TYPER font is shown in Fig.2.7, while OCR-A characters are shown in Fig.2.6 already. Some difficult points by the reading of normal typewriter fonts are that the relative heights of characters are not equal and that the relative vertical positions of these characters are not equal. Some of the examples are shown in Fig. 2.8. Therefore the proper base line is very difficult to be found. The design of characteristic feature logics are done to accept as many variations of fonts as possible.

2.2 Recognition Principle

2.2.1 Fundamental Principle of Character Recognition

The input pattern is in the form of 16 channels of pulses (named from top to bottom A,B,C,D,E,F,G,H,h,g,f,e,d,c,b,a), which change as the original pattern passes through the input part. This is schematically shown in Fig. 2.9. Here several kinds of logical conditions are provided for among these 16 channels, which produce signals as long as the logical conditions are satisfied among these 16 channels as a pattern passes through the input part. These logical conditions, which are named characteristic feature logics, are set up corresponding to several vertical sections of characters. For example, three vertical sections x_1, x_2 and x_3 are selected for the character "2" as shown in Fig. 2.10, which are thought of as

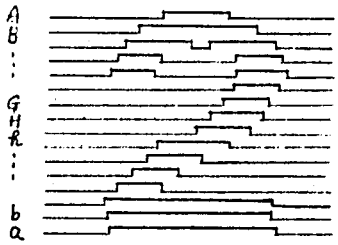


Fig. 2.9 16 channels of input signals for character "2".

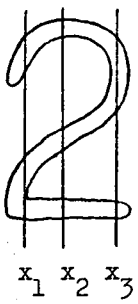


Fig. 2.10 Three vertical sections for character "2".

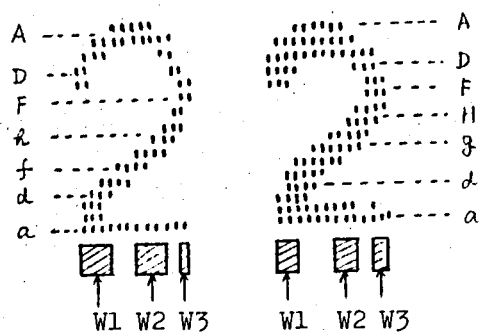


Fig. 2.11 Sampled pattern of "2". The shaded areas satisfy the logical conditions.

representing the character fairly well. At these vertical sections the following logical conditions can be written down among the 16 channels.

$$x_1 : W = \overline{A}\,(B + C + D)\,\overline{G}\,\overline{H}\,\overline{h}\,\overline{g}\,d\,c\,(b + a)$$

$$x_2 : W = (A + B + C)\,\overline{E}\,\overline{F}\,\overline{G}\,(H + h + g)\,\overline{e}\,\overline{d}\,(b + a)$$

$$x_3 : W = \overline{A}\,E\,(D + F)\,(H + h + g)\,\overline{f}\,\overline{e}\,\overline{d}\,(c + b + a)$$

For the sampled pattern "2" of Fig. 2.11 the intervals satisfied by these characteristic feature logics are shown for the illustration, although the input patterns are not digitized in time axis. These three conditions are determined to fit to a character which is shifted one channel up or down, to be mutually independent, and not to be strict as far as there arises no confusion in the recognition with other characters. When a character passes through the input part, there are three pulse outputs at asynchronous time points t_1, t_2, t_3, each corresponding to the sections x_1, x_2 and x_3. These pulses continue as long as the portions of a pattern satisfy the conditions. The recognition is done by asynchronous sequential logical circuits, which detect only the sequence of appearance of these three conditions in this order as a character goes by under the input part, so that the absolute pulse width and time intervals between these pulses have no effect on the recognition. Therefore the scanning speed may be fast, slow or even may vary. This is a very profitable point for the machine, because it does not demand the high precision on the scanning mechanism.

Three vertical sections are considered to be enough for the recognition of numerals and capital English alphabet. If more than three sections are required, three or more memory flip-flops are necessary for the sequence detection of the sections. The functions to be set up for these sections must become more minute in this case, so that the error will increase when there is noise or small deviations on the character.

Therefore for the characters which are hard to be distinguished by three vertical sections, it might be better to introduce some auxiliary principles rather than to increase the number of vertical sections. Actually the principle mentioned above is not enough when there are many similar characters to be recognized. For example, by the character pairs 2 and 8, I and K, C and O and so on, which are shown in Fig. 2.12, the former characters can be considered as embedded in the latter characters.

In these cases the latter patterns are recognized as both the former and the latter by the recognition principle mentioned in the above section.

A solution is that when there are two simultaneous recognition results, which are previously known such as the pairs 2 and 8, I and K, C and O, etc., one output corresponding to the embedded pattern is suppressed by the other, which is to be output.

Another solution is to detect a characteristic feature logic which is satisfied by a character and which is not satisfied by the other.

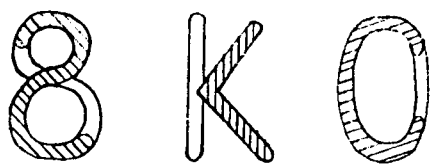


Fig. 2.12 Examples of patterns embedded in another patterns. 2 and 8, 1 and K, C and O.

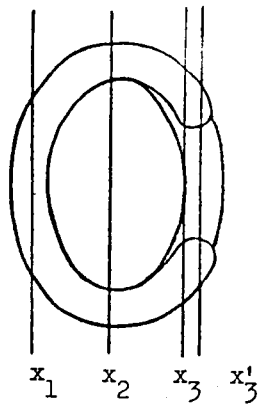


Fig. 2.13 Distinction between C and O. Logic condition at x_3' does not appear for C.

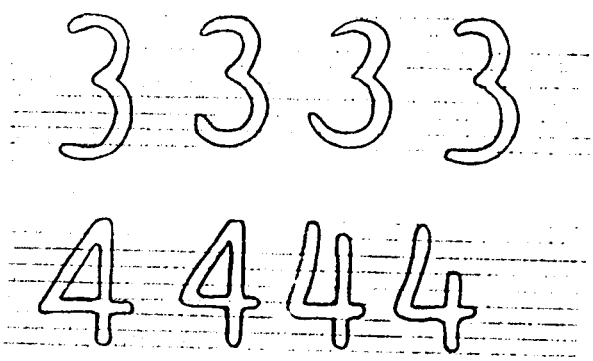


Fig. 2.14 Several different fonts which are taken into consideration for the design of characteristic feature logics.

In the case of C and O, a different point between these is the vertical section x_3', which is shown in Fig. 2.13, so that if the characteristic feature logic set up at x_3' appeared after the logics at x_1, x_2, and x_3 are satisfied, the recognition output is suppressed.

2.2.2 Characteristic Feature Logics

The characters we want to recognize are numerals and capital letters of English alphabet, so that three vertical sections are enough for the recognition of each character. The design principle of each characteristic feature logics are,

(1) the first characteristic feature logic and the second, then the second characteristic feature logic and the third, are to be independent each other,

(2) each characteristic feature logic for a section of a pattern is to be satisfied still by the condition that the pattern is shifted one channel up or down,

(3) each characteristic feature logic must be determined to be satisfied still by the condition that the sections of a pattern may have certain amount of extraneous ink or deficiencies,

(4) each characteristic feature logic should be strict enough to recognize each character without confusion with another,

(5) each characteristic feature logic is to be determined to work for a slight change of a pattern or for multifont characters,

(6) it should be considered as far as possible that even if

one or two channels have troubles and no signal outputs are obtained, characteristic feature logics work well.

The determination of characteristic feature logics is very difficult especially when the number of these is many. There is no theoretical method for the determination of the best sections and their best logical functions. Therefore the determination has been done empirically, but the effectiveness of these logical conditions has been tested by digital computer simulation of the system in several conditions, which will be mentioned in Chapter 4.

For example the following characteristic feature logics are provided for characters 3 and 4.

For character "3",

$$W1 = (A + B + C)\ \bar{E}\ \bar{F}\ \bar{G}\ \bar{H}\ \bar{h}\ \bar{g}\ \bar{f}\ \bar{e}\ \bar{d}\ \bar{c}$$

$$W2 = (A + B + C)\ (\bar{D} + \bar{F})\ \bar{E}\ (G + H + h)\ \bar{f}\ \bar{e}\ \bar{d}$$

$$W3 = \bar{A}\ \bar{E}\ (D + F)\ (\bar{G} + \bar{H} + \bar{h})\ (g + f)\ (e + d + c)$$

For character "4",

$$W1 = \bar{A}\ \bar{B}\ \bar{C}\ \bar{D}\ \bar{E}\ \bar{F}\ g\ f\ (h + e)\ \bar{b}\ \bar{a}$$

$$W2 = (C + D + E)\ \bar{G}\ \bar{H}\ \bar{h}\ (f + e + d)\ \bar{b}\ \bar{a}$$

$$W3 = (F + G)\ (H + h)\ (g + f)\ (e + d)\ (c + b)$$

These logics work well for several characters such as shown in Fig. 2.14.

About 50 such characteristic feature logics are provided for the recognition of 37 characters mentioned in 2.1.3. These are listed in Table 2.1.

Minimization of these characteristic feature logics is an interesting problem, because here is another factor i.e. se-

Table 2.1 Characteristic feature logics set up for 37 characters.

Q1 = $\overline{A}\ \overline{B}\ F\ G\ H\ h\ g\ \overline{b}\ \overline{a}$

Q2 = $(A + B + C)\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{f}\ \overline{e}\ (c + b + a)$

Q3 = $\begin{matrix}(B + C)\ (D + E)\ (F + G)\ (H + h)\ (g + f)\ (e + d) \\ (c + b)\end{matrix}$

Q4 = $\overline{A}\ (B + C + D)\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ d\ c\ (b + a)$

Q5 = $(A + B + C)\ \overline{E}\ \overline{F}\ \overline{G}\ (H + h + g)\ \overline{e}\ \overline{d}\ (b + a)$

Q6 = $\overline{A}\ E\ (D + F)\ (H + h + g)\ \overline{f}\ \overline{e}\ \overline{d}\ (c + b + a)$

Q7 = $(A + B + C)\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}$

Q8 = $(A + B + C)\ (\overline{D} + \overline{F})\ \overline{E}\ (G + H + h)\ \overline{f}\ \overline{e}\ \overline{d}$

Q9 = $\overline{A}\ E\ (D + F)\ (\overline{G} + \overline{H} + \overline{h})\ (g + f)\ (e + d + c)$

Q10 = $(A + B)\ \overline{E}\ (H + h + g)\ (d + c + b)$

Q11 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ g\ f\ (h + e)\ \overline{b}\ \overline{a}$

Q12 = $(C + D + E)\ (\overline{h} + \overline{f})\ \overline{g}\ (e + d + c)\ \overline{a}$

Q13 = $(F + G)\ (H + h)\ (g + f)\ (e + d)\ (c + b)$

Q14 = $B\ C\ D\ E\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}$

Q15 = $(A + B)\ (\overline{C} + \overline{E})\ \overline{D}\ (F + G\ H)\ \overline{f}\ \overline{e}\ (c + b + a)$

Q16 = $(A + B)\ \overline{D}\ \overline{E}\ g\ f\ e\ \overline{a}$

Q17 = $\overline{A}\ \overline{B}\ \overline{C}\ G\ H\ h\ g\ f\ e\ \overline{a}$

Q18 = $(A + B + C)\ (\overline{D} + \overline{F})\ \overline{E}\ (G + H + h)\ \overline{f}\ \overline{e}\ \overline{d}\ (b + a)$

Q19 = $\overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ e\ (f + d)\ \overline{a}$

Q20 = $\overline{A}\ \overline{B}\ E\ (D + F)\ (\overline{H} + \overline{h})\ (g + f)\ (\overline{e} + \overline{d} + \overline{c})\ (b + a)$

Q21 = $(A + B + C)\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ e\ d\ c\ b$

Q22 = $B\ C\ D\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q23 = $(A + B)\ (\overline{D} + \overline{F})\ \overline{E}\ (G + H + h)\ (\overline{f} + \overline{e} + \overline{d})\ (b + a)$

Q24 = $\overline{A}\ E\ F\ G\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}$

Q25 = $(A + B)\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ (g + f + e)\ (\overline{d} + \overline{b})\ \overline{c}$

Q26 = $\overline{A}\ \overline{B}\ F\ G\ H\ h\ g\ e\ \overline{b}\ \overline{a}$

Q27 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ d\ c\ b$

Q28 = $B\ C\ \overline{G}\ \overline{H}\ (g + f + e)\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q29 = $\overline{A}\ \overline{B}\ (D + F)\ E\ (\overline{G} + \overline{H} + \overline{h})\ (g + f)\ (c + b)$

Q30 = $(C + D + E)\ \overline{H}\ \overline{h}\ (\overline{G} + \overline{g})\ (e + d + c)$

Table 2.1 (continued)

Q31 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ (H + h + g)\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q32 = $(B + C + D)\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ \overline{f}\ (d + c + b)$

Q33 = $(A + B + C)\ (\overline{D} + \overline{F})\ \overline{E}\ (H + h + g)\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q34 = $\overline{A}\ D\ E\ F\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q35 = $\overline{A}\ D\ E\ (\overline{h} + \overline{f})\ \overline{g}\ (d + b)\ c$

Q36 = $D\ E\ (\overline{H} + \overline{g})\ \overline{h}\ c$

Q37 = $(C + E)\ D\ (\overline{F} + \overline{H})\ \overline{G}\ (g + f)\ (c + b)$

Q38 = $(B + C)\ (D + E)\ (F + G)\ (H + h)\ (g + f)$

Q39 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{f}\ \overline{e}\ (c + b + a)$

Q40 = $B\ C\ D\ \overline{h}\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ a$

Q41 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ d\ c\ b$

Q42 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ h\ g\ f\ d\ c\ b$

Q43 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ H\ h\ (G + g)\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q44 = $\overline{A}\ \overline{B}\ D\ (C + E)\ \overline{H}\ \overline{h}\ d\ (e + c)\ \overline{b}\ \overline{a}$

Q45 = $(A + B)\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{f}\ \overline{e}\ \overline{d}\ (b + a)$

Q46 = $(B + C + D)\ E\ F\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q47 = $(B + C)\ D\ E\ (F + G)\ \overline{f}\ \overline{e}\ (c + b + a)$

Q48 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ (G + H + h + g + f)\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q49 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ (h + g + f)\ (\overline{d} + \overline{e} + \overline{c})\ (b + a)$

Q50 = $(A + B)\ (\overline{D} + \overline{C} + \overline{E})\ (F + G + H)\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q51 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ e\ d\ c$

Q52 = $(B + C + D)\ E\ (F + G)\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q53 = $(A + B + C)\ \overline{D}\ \overline{E}\ (g + f)\ (e + d)\ (c + b)\ (H + h)$

quential combination of three logical functions, which doesnot appear in the ordinary minimization problem.

For the mere distinction of 37 different kinds in hypothetic ideal case, 4 logical functions are enough, because $4^3 = 64$. However the characteristic feature logics of the actual characters are not so equally distributed that this is rather an absurd comparison. The reason why the minimization was not tried is as follows.

(1) If the minimization is done too far, patterns which are completely different from the originally intended ones may be recognized as the same, because there is so much freedom of variation.

(2) If the minimization is done too far, for the patterns which are very noisy the sequence detection may become inaccurate and there is a possibility that many recognition outputs may appear simultaneously.

(3) When new characters are to be added to the machine, minimization is to be done once again. This incurs a great trouble, because all the logics are to be changed for the addition of only one character. By the characteristic feature logics adopted here, the addition of a new character has no effect for the recognition of the other characters. This is the most advantageous point of this machine. Actually the design of this machine was initially done only for the NEAC TYPER fonts. But ISO recommended the adoption of OCR-A type font for the optical character recognition, so that OCR-A font was

added to the system without any trouble. In this case few characteristic feature logics were added without any alteration of the logics already existed. Many of the logics for NEAC TYPER font could be utilized in common for OCR-A type font.

(4) Minimization of the logical functions of 16 variables will take tremendous time even by using high speed computers.

2.2.3 Recognition Logics

For the detection of the sequence of appearance of three characteristic feature logics, an asynchronous sequential logic circuit is provided for, which is composed of two memory flip-flop circuits F1 and F2. Initially the state (F1, F2) of the two flip-flops is set to (0, 0), which corresponds to the non-existence of a character under the input part. When the first characteristic feature logic W1 is satisfied, the flip-flops are turned to the state (0, 1). Next when the second characteristic feature logic W2 is satisfied at the internal state (0, 1), the state is turned into (1, 1). Next by the appearance of the third characteristic feature logic W3, the state is turned into (1, 0). This final internal state (1, 0) is maintained until the character passes completely through the input part. At this point when the logical condition

$$\bar{Y} = \overline{A + B + \cdots + G + H + h + g + \cdots + b + a} = 1$$

is satisfied, the internal state (F1, F2) is examined. For the characters whose internal states (F1, F2) are (1, 0), the

recognition outputs are produced. For the other sequences of appearance of W1, W2, and W3 the state (F1, F2) is designed not to reach to the final stage (1, 0). The whole process of this is represented by the state diagram shown in Fig. 2.15.

F1 F2	W1	W2	W3	W0	N
0 0	01	00	00	00	00
0 1	01	11	01	01	00
1 1	11	11	10	11	00
1 0	10	10	10	10	00

(1) unsymmetric characters

F1 F2	W1	W2	W0	N
0 0	01	00	00	00
0 1	01	11	01	00
1 1	10	11	11	00
1 0	10	10	10	00

(2) symmetric characters

Fig. 2.15 State diagram of recognition circuits.

The state diagram of symmetric characters such as 0, 8, A, U, V, X, Y etc. is different from that of the unsymmetric characters. For the symmetric characters the first and the third characteristic feature logics are the same. W1, W2, and W3 represent the first, second and the third characteristic feature logics respectively. N is the reset pulse of all the circuit, and appears 600 μsec. after the character has passed completely through the input part. The pulse width is about 300 μsec. W0 is the complement of W1 + W2 + W3 + N. From this state diagram the following logical equations are obtained, where

$$W0 = \overline{W1 + W2 + W3 + N}$$

for the unsymmetric characters, and

$$W0 = \overline{W1 + W2 + N}$$

for the symmetric characters.

(1) Unsymmetric characters :

$$F1 = \overline{N} \cdot F1 + W2 \cdot F2$$

$$F2 = W1 \cdot \overline{F1} + \overline{W3} \cdot \overline{N} \cdot F2 + W3 \cdot \overline{F1} \cdot F2$$

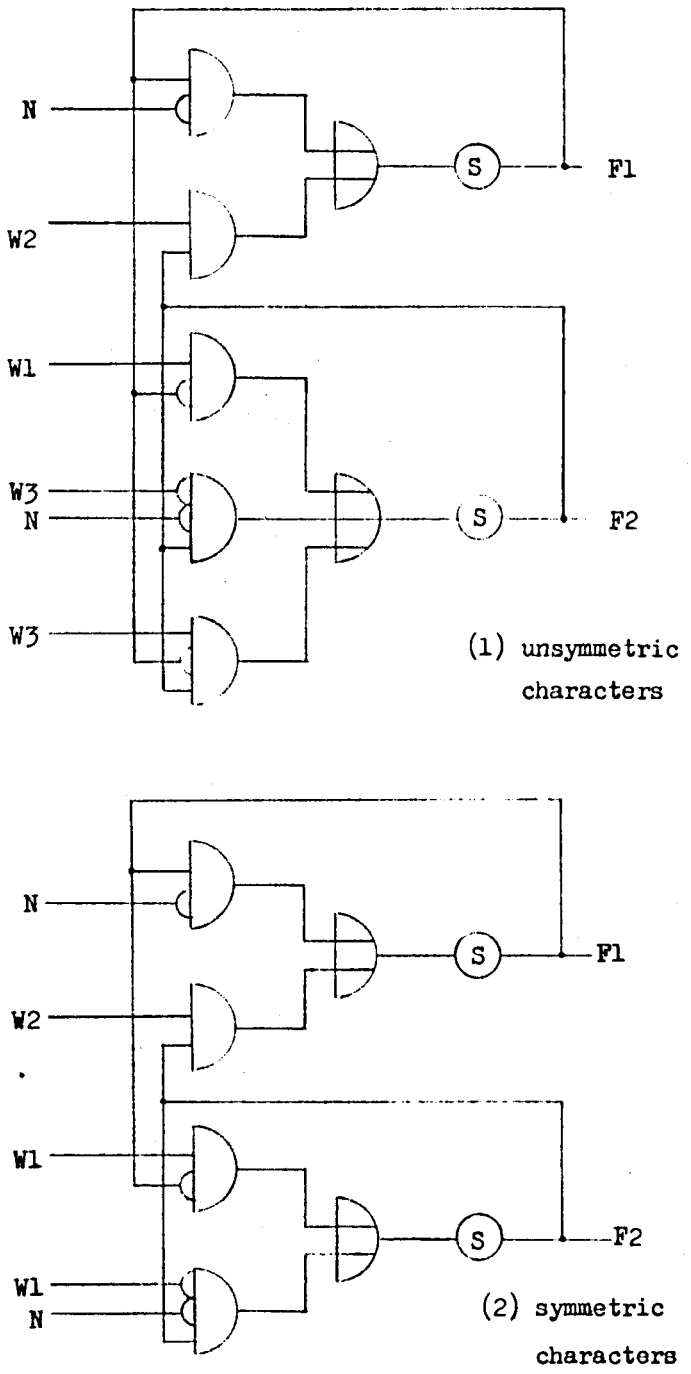


Fig. 2.16 Circuit diagram of recognition logics.

(2) Symmetric characters :

$$F1 = \overline{N} \cdot F1 + W2 \cdot F2$$

$$F2 = W1 \cdot \overline{F1} + \overline{W1} \cdot \overline{N} \cdot F2$$

A term $W1 \cdot W3 \cdot F2$ is omitted from the equation of F2 for unsymmetric characters, because W1 and W3 are often independent, and even if they are not independent the contribution of the term is very small.

The logic circuit diagrams of these equations are shown in Fig. 2.16. The signal obtained from the diode gates is regenerated by the circuit S, which is a slicer and is explained in 2.5.3.

2.2.4 Inhibition Logics

When a pattern is just embedded in another pattern such as C is embedded in O, which is shown in Fig. 2.12 or when a sequence of characteristic feature logics of a pattern is realized by some other patterns, there appear more than one recognition outputs. In the case of C and O, the difference between these is the vertical section x_3' as shown in Fig.2.13, so that if the characteristic feature logic prepared for x_3' appeared after the flip-flops (F1, F2) for the character C become (1, 0), the output of C is suppressed. In such cases like this, where a certain characteristic feature logic appeared at a certain state of the flip-flops, the inhibition of recognition output is performed. For this purpose another memory flip-flop Fz is provided.

The condition of the inhibition logic for Fz differs for each character but for C the state diagram shown in Fig. 2.17 can be written down.

F1	F2	Fz	W1	W2	W3	W3'	W0	N
0	0	0	010	000	000	000	000	000
0	1	0	010	110	010	010	010	000
1	1	0	110	110	100	110	110	000
1	0	0	100	100	100	ØØ1	100	000
Ø	Ø	1	ØØ1	ØØ1	ØØ1	ØØ1	ØØ1	000

Fig. 2.17 State diagram for the recognition of character C.

From this state diagram the following logical equations are derived.

For C

$$F1 = \overline{N} \cdot F1 + W2 \cdot F2$$

$$F2 = W1 \cdot \overline{F1} + \overline{W3} \cdot \overline{N} \cdot F2 + W3 \cdot \overline{F1} \cdot F2$$

$$Fz = \overline{N} \cdot Fz + W3' \cdot F1 \cdot \overline{F2}$$

where

$$W1 = \overline{A}\ \overline{B}\ F\ G\ H\ h\ g\ f\ \overline{b}\ \overline{a}$$

$$W2 = (A + B + C)\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ \overline{f}\ \overline{e}\ (c + b + a)$$

$$W3 = (C + D + E)\ \overline{H}\ \overline{h}\ (\overline{G} + \overline{g})\ (e + d + c)$$

Next the case between 2 and 8 is discussed. The characteristic feature logics of 2 and 8 are,

for 2

$$W1 = \overline{A}\ (B + C + D)\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ (e + d + c)\ (b + a)$$

$$W2 = (A + B + C)\ \overline{E}\ \overline{F}\ \overline{G}\ (H + h + g)\ \overline{e}\ \overline{d}\ (b + a)$$

$$W3 = \overline{A}\ D\ E\ F\ \overline{g}\ \overline{f}\ \overline{e}\ (c + b + a)$$

for 8

$$W1 = \overline{A}\ (D + F)\ E\ (\overline{G} + \overline{H} + \overline{h})\ (g + f)\ (e + d + c)$$

$$W2 = (A + B)\ (\overline{D} + \overline{F})\ \overline{E}\ (G + H + h)\ (\overline{f} + \overline{d})\ \overline{e}\ (b + a)$$

$$W3 = W1$$

There are intersections between W1 of 2 and W1 of 8,

and between W2 of 2 and W2 of 8. Also there is a possibility that a portion of character 8 satisfies the logic W3 of 2 after the conditions W1 and W2 of 8 are satisfied, whereas there is little possibility of the satisfaction of the logic W3 of 8 by the right half of input pattern 2. This situation is shown in Fig. 2.18. In this case when the input is a character 8 there are outputs 8 and 2. Therefore the inhibition condition for the output 2 in this case is the appearance of W3 of 8 at the internal state F1 = 1 of the character 2. So

$$Fz(2) = Fz(2)\cdot\overline{N} + F1(2)\cdot W3(8)$$

The characters in the brackets indicate that the corresponding flip-flops or characteristic feature logics belong to their respective characters.

These situations may arise in many character pairs and for the logical design we have to investigate all these situations. This is very troublesome task, so that another not complete but very convenient method is adopted.

The logic W3(= W1) of 8 is very similar to the logic W1 of 2, so the result will be the same even if we adopt the logic W1 of 2 for the above inhibition logic instead of W3 of 8.

To extend this idea to the whole interval of signal existence, we may design the inhibition logics to be activated when W2 or W3 appears before W1, when W3 appears before W2, when W1 appears after W2 or W3, and when W1 or W2 appears after W3. The state diagram of this function is shown in Fig. 2.19. From this the following logic functions are derived. For unsymmetric characters,

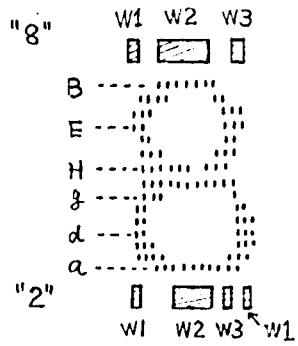


Fig. 2.18 Confusion of 2 and 8. W1, W2, and W3 in the upper part are the characteristic feature logics for 8, while those in the lower part are for 2.

F1	F2	Fz	W1	W2	W3	W0	N
0	0	0	010	ØØ1	ØØ1	000	000
0	1	0	010	110	ØØ1	010	000
1	1	0	ØØ1	110	100	110	000
1	0	0	ØØ1	ØØ1	100	100	000
Ø	Ø	1	ØØ1	ØØ1	ØØ1	ØØ1	000

(1) unsymmetric characters

F1	F2	Fz	W1	W2	W0	N
0	0	0	010	ØØ1	000	000
0	1	0	010	110	010	000
1	1	0	100	110	110	000
1	0	0	100	ØØ1	100	000

(2) symmetric characters

Fig. 2.19 State diagram for the recognition logics with inhibition memory Fz.

$$F1 = W2 + \overline{N} \cdot F1$$

$$F2 = W1 + \overline{W3} \cdot \overline{N} \cdot F2$$

$$Fz = W1 \cdot F1 + W2 \cdot \overline{F2} + W3 \cdot \overline{F1} + \overline{N} \cdot Fz$$

For symmetric characters,

$$F1 = W2 + \overline{N} \cdot F1$$

$$F2 = W1 \cdot \overline{F1} + \overline{W1} \cdot \overline{N} \cdot F2$$

$$Fz = W2 \cdot \overline{F2} + \overline{N} \cdot Fz$$

By these recognition logics the design of characteristic feature logics is very easy because references to other characters are seldom required. And these recognition logics have been working fairly well. Moreover this is advantageous because the circuits of the recognition are the same for all the characters.

The connection between the characteristic feature logics and the recognition logics has been done by the plug-in boad. By the adoption of this plug-in boad system the change of the characteristic feature logics can be done with minimum alteration of the system. The machine has 7 and 4 recognition circuits for unsymmetric and symmetric characters respectively.

The characteristic feature logics W1, W2, W3 for the recognition of characters are shown in Table 2.2. For the characters C, S, 1, special circuits are used, for which no plug-in connections are used.

2.2.5 Recognition of Special Characters

For the characters 1, L, etc. we can not take three characteristic feature logics. Special considerations are requir-

Table 2.2 Recognition logics.

	W1	W2	W3	
0	Q1	Q2		
1	Q3			Inhibition by 3,4,8,9,B,C, K,N,P,R,U.
2	Q4	Q5	Q6	
3	Q7	Q8	Q9	
4	Q11	Q12	Q3	
5	Q14	Q15	Q19	
6	Q1	Q18	Q19	
7	Q7	Q21		
8	Q9	Q23		
9	Q24	Q25	Q17	
<	Q43	Q44	Q45	Inhibition by Q3
>	Q45	Q44	Q43	
D	Q3	Q2		
ı	Q3			The same as "1"
ƨ	Q53	Q23	Q47	
Ɜ	Q2	Q23	Q3	
ч	Q46	Q48	Q13	
S	Q47	Q23	Q53	
b	Q3	Q49	Q51	
?	Q7	Q21	Q46	
8	Q42	Q23		
٩	Q52	Q50	Q3	
ſ	Q51	Q3	Q14	
Ч	Q52	Q42		
⊓	Q42	Q48	Q3	

Table 2.2 (continued)

	W1	W2	W3	
A	Q27	Q28		
B	Q3	Q23	Q29	
C	Q1	Q2	Q30	Special inhibition circuit
K	Q3	Q31	Q32	
N	Q3	Q31		
P	Q3	Q33	Q34	
R	Q3	Q33	Q35	
S	Q36	Q23	Q37	Special inhibition circuit
U	Q38	Q39		
V	Q40	Q41		
X	Q32	Q31		
Y	Q22	Q42		

ed for the recognition of these characters, which are compatible with the principle of this machine.

The characteristic feature logic provided for the vertical line is

$$Q3 = (B + C)\ (D + E)\ (F + G)\ (H + h)\ (g + f)\ (e + d)\ (c + b).$$

This vertical line is contained in many other characters as their portions, such as, 4, B, K, N, P, R etc.. Also there is a possibility that a vertical line exists in characters such as 3, 4, 8, 9, C, U, etc., when some distortions or noise may exist on them. Therefore the recognition of "1" is done by the appearance of the above logic Q3 and there is no recognition output from any of the characters having the vertical lines. The timing of the recognition of 1 therefore delays a little behind that of the recognition of normal characters. This delay time is adjusted about 300 μsec.

The logic circuit of the recognition of 1 is shown in Fig. 2.20. The characteristic feature logic for 1 is connected to line a and its occurrance is memorized in circuit b. The recognition outputs of characters B, K, N, P, R, etc., all having the vertical lines, are connected to the AND gate c. The output of this OR gate inhibits the output of the circuit b. The recognition result is obtained from the circuit d.

By the recognition principle the length of the horizontal lines can not be detected. Therefore characters L, F, T, etc. can not be distinguished from character 1 with serif or with edge irregularities, so that some modifications are required. For the special characters such as L and F to be distinguished

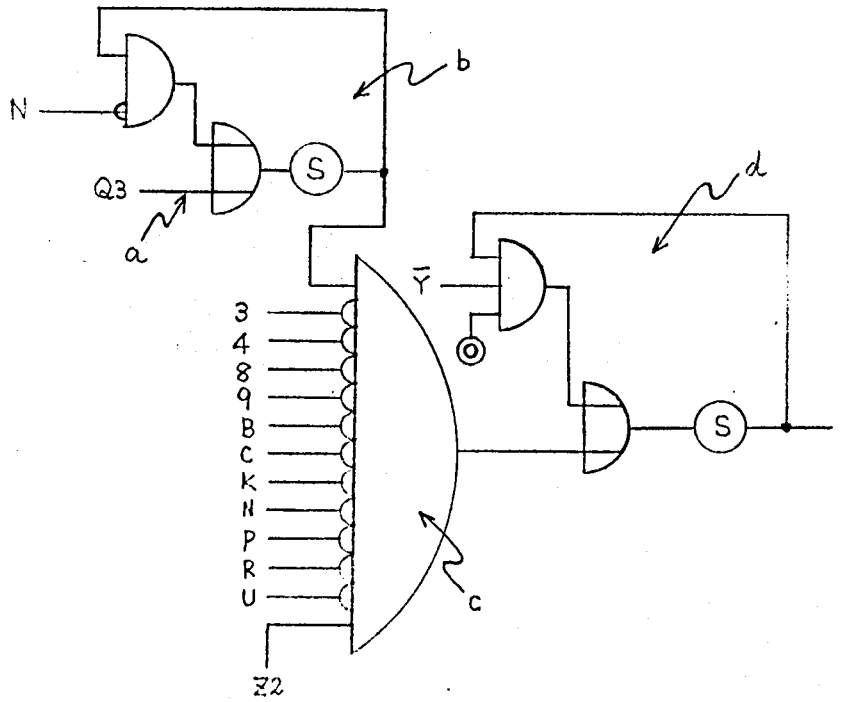


Fig. 2.20 Recognition circuit of character 1.

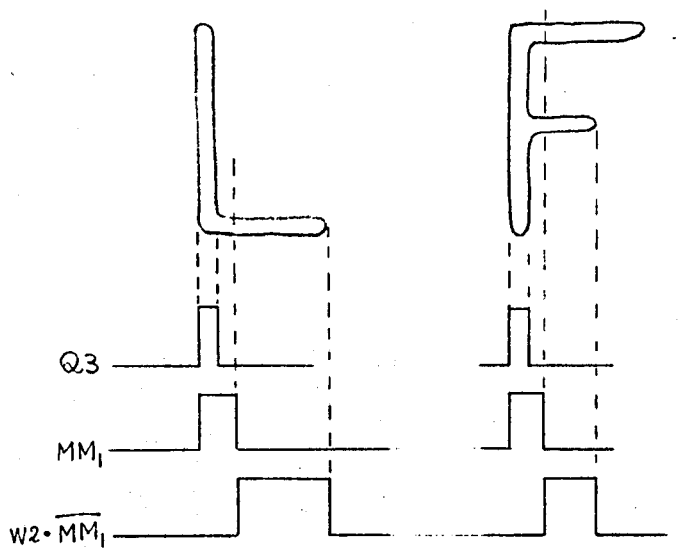


Fig. 2.21 Recognition of L and F. MM_1 is an output of one-shot multivibrator which is triggered by Q3.

from numeral 1, certain time interval is introduced after the characteristic feature logic for 1 is satisfied. The second characteristic feature logics for L and F are to be examined after this time interval and if at that time the conditions are still satisfied, the recognitions for L and F are output, otherwise the recognition is for 1. This is schematically illustrated in Fig. 2.21. This time interval is produced by one-shot flip-flop triggered by the pulse of the logic for 1, and the interval can be adjusted to absorb some irregularities of scanning speed. The recognition circuit for L or F is shown in Fig. 2.22, in which

for L $W2 = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}\,\overline{F}\,\overline{G}\,\overline{H}\,\overline{h}\,\overline{f}\,\overline{g}\,\overline{e}\,(c + b + a)$

for F $W2 = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}\,\overline{F}\,(G + H + h + g + f)\,\overline{d}\,\overline{c}\,\overline{b}\,\overline{a}$

The same treatment is required for W2 of character E. If the middle horizontal line of E is the same length of upper and lower horizontal lines, the recognition circuit is the same as that of L or F. If the middle line is shorter, the circuit is similar to those of ordinary characters, and an only difference is that W2 is detected after a certain time interval from the end of a vertical line.

For the recognition of character T, two such time intervals are to be introduced such as shown in Fig. 2.23. The first time interval is triggered by the logical function

$W1 = (A + B + C)\,\overline{E}\,\overline{F}\,\overline{G}\,\overline{H}\,\overline{h}\,\overline{g}\,\overline{f}\,\overline{e}\,\overline{d}\,\overline{c}\,\overline{b}\,\overline{a}$

and this works for the separation of W1 from the vertical line in the center. The second time interval is triggered by the vertical line and is for the separation of the second W1 from

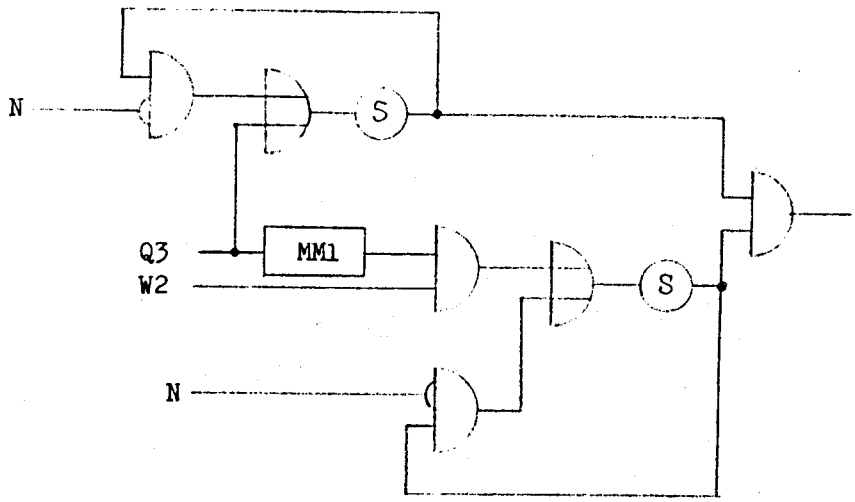


Fig.2.22 Recognition circuit for L or F.

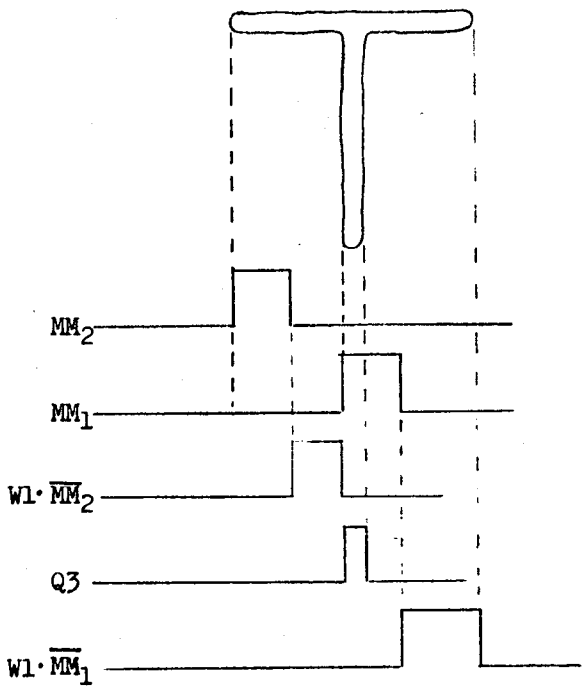


Fig. 2.23 Recognition of character T. MM_2 is triggered by W1.

the vertical line.

For the recognition of M and W, and for the separation of H and N three vertical sections are not enough. However the sections are not increased by the reason mentioned in 2.2.1. Instead, two counters are provided for the detection of the number of horizontal crossings of a character. The counter A is triggered by the output of logical function

$$Ca = F(E + G)$$

and the counter B is triggered by

$$Cb = f(e + g)$$

Then the counter outputs for M, W, H and N are as shown in Table 2.3, which are enough for the recognition of each character.

		M	W	H	N
upper counter	Ca	4(2)	3	2	3
lower counter	Cb	4(3)	2	2	3

Table 2.3 Number of cross sections of the upper and lower parts of M, W, H, and N.

These two counters can also be utilized for the check of the recognition result for each character, and thus serve for the higher accuracy of the system.

2.3 Input Part

2.3.1 Input Device

When a pattern recognition system is considered as a commercial machine, an essential point is what kind of input device is used. This closely relates to the recognition principle, as well as the performance of the system. Among the

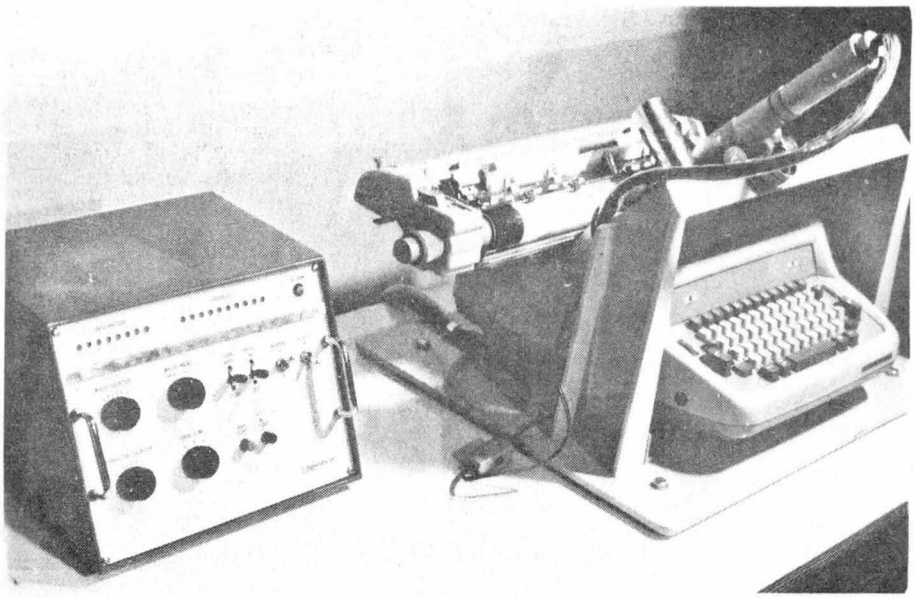


Fig. 2.24 Input scanner and control panel.

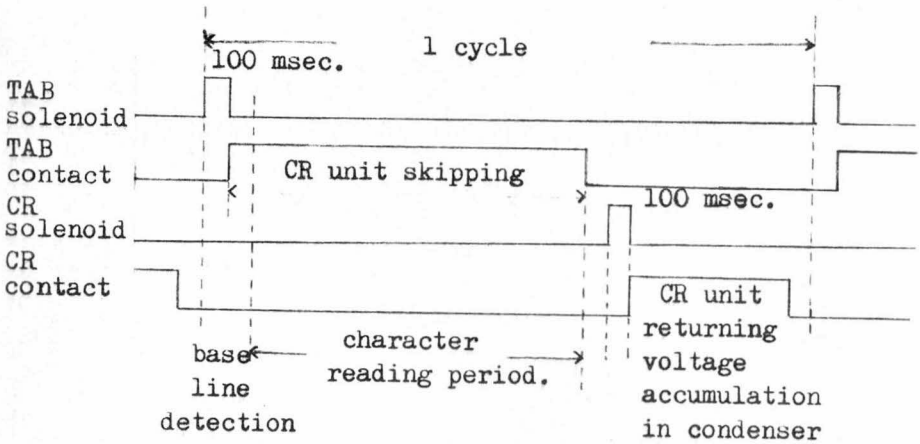


Fig. 2.25 Timing of the carriage movement.

problems to be considered by the input device, the mechanics of feeding documents and the scanning method are the two important problems. There are several methods of card and page feed. Also there are many scanning devices such as flying spot scanner, rotating disk and so on. The method adopted here is an electric typewriter and photo transistor.

A document paper is inserted to the typewriter carriage and this carriage moves left and right by the tabulator (TAB) and carriage return (CR) actions. The reading head is attached at right angles to the typewriter platen. Therefore a paper is scanned horizontally. The light is projected to the surface of a paper, whose reflected light carries the optical image of patterns. This optical image is enlarged about three times by a lens system and is converted to electric signals by photo transistors. This input device is shown in Fig. 2. 24.

The carriage of an electric typewriter is operated by the pulses schematically shown in Fig. 2.25. By the inertia of the carriage, the pulses for TAB and CR solenoids are to be applied with adequate intervals of more than 100 mili-second after the CR and TAB actions end and the respective relay contacts open. The line feed is done at the beginning of the carriage return action.

Therefore the trace of the movement of the scanning head on a paper is shown schematically in Fig. 2.26.

A part of the inserted paper under the scanning head is lighted up by a lamp which operates in D.C. in 8 volts, 50

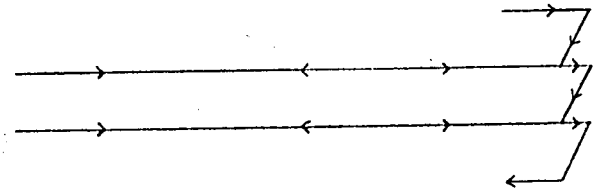


Fig. 2.26 Trace of the carriage movement.

watts. If AC source is used the intensity of light fluctuates in 60 cycles per second and is not preferable. The photo transistor used here has the peak of sensitivity around the 10000--13000 Å , so that an incandescent lamp is suitable.

The reflected light images of patterns are enlarged to 8 mm height by a lens system as shown in Fig. 2.27. The typed characters are usually 2.7--3.0 mm of height, so that the images at the focus plane are magnified about three times. On this focus plane 22 glass tubes are attached as shown in Fig. 2.27. These are aligned vertically to the optical images of input characters. The diameter of a glass tube is 0.5 mm, so that 16 glass tubes cover the character height. The other end of each glass tube is attached to the head of a photo transistor PD-6. Glass tube is used because the diameter of PD-6 is 2 mm and can not be put in a row within such restricted area. The loss of light by the reflection from both ends of the tube and by the leakage from the side is not measured, but it seems that the loss can be neglected.

The cross section of the glass tube is made circle, but the following considerations suggest the rectangular cross

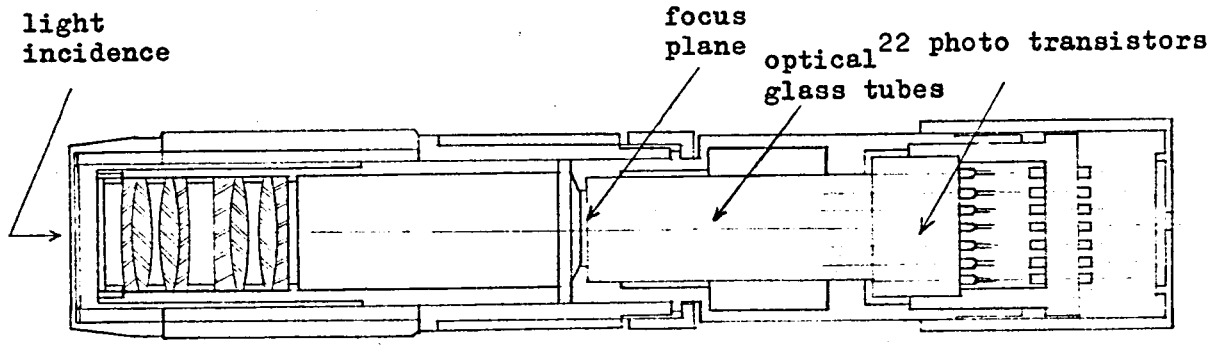


Fig. 2.27 Lens system which enlarges three times the optical image.

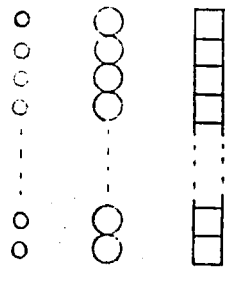


Fig. 2.28 Variety of input sections.

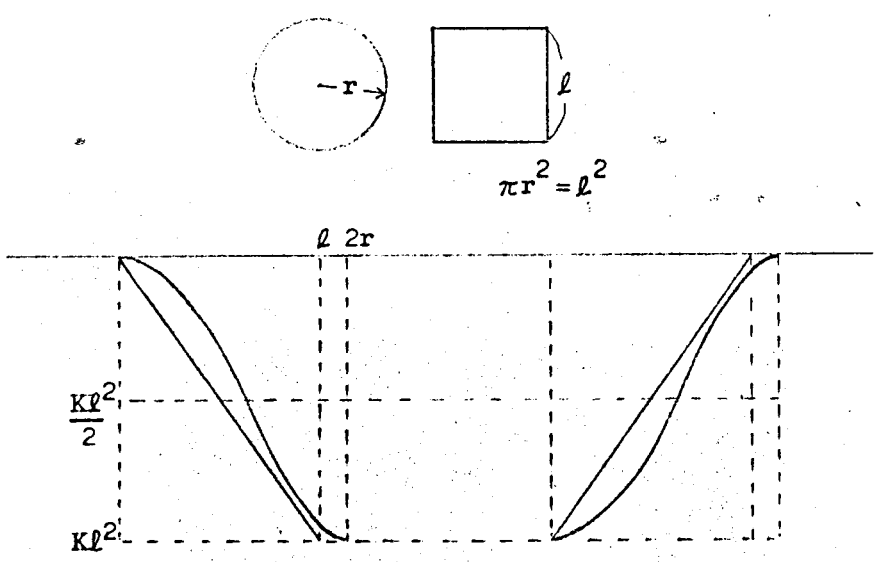


Fig. 2.29 Light quantity curves when a vertical line comes in to the above cross sections.

section is better. The factors to be considered are :

(1) The loss of the input light must be as little as possible.

(2) The inclination of a vertical line must be detected as sharply as possible.

Let us compare the three input sections shown in Fig. 2. 28. The input section A is apparently unsuitable by the reason (1) mentioned above. If the cross section of a rectangle is supposed to be a square and its area is made equal to the area of the circle, the ratio of the side length of a square and the diameter of a circle is 1 : 0.886. The light quantity when a vertical line comes in to the cross section varies as shown in Fig. 2.29. If the slice level for 1 and 0 is set at 50% of the maximum light quantity, the gradient of the curve of the circle is larger than that of the square, that is, the sensitivity for the small shift of the slice level is better by the circle. If the side length of a square is made equal to the diameter of a circle, the light quantity curves are as

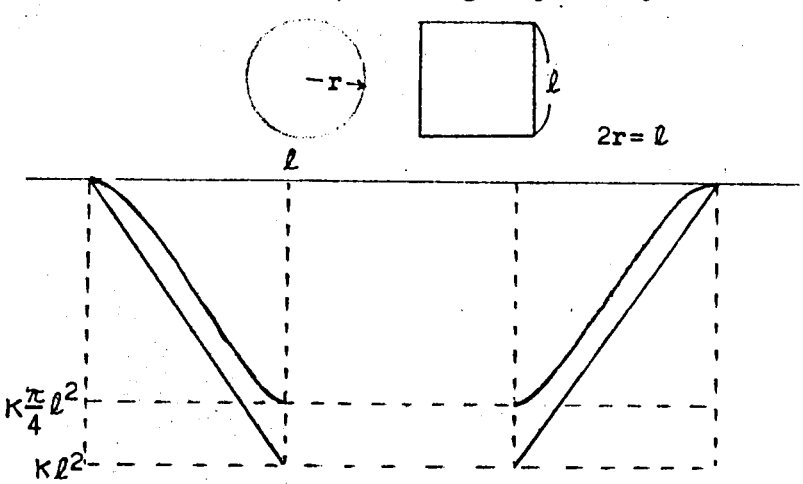


Fig. 2.30 Light quantity curves when a vertical line comes in to the above cross sections.

shown in Fig. 2.30. In this case the light quantity of a circle is far little than that of the square.

Next let us compare the two input sections, square and rectangle. For the simplicity of discussion a rectangle is supposed to be the form one side is twice the other side in length. If both areas are made equal and if a vertical line slightly inclined comes in, the light intensity curves of both sections are as shown in Fig. 2.31. The gradient of the curve of rectangle is twice that of the square.

From this discussion it is concluded that the square or rectangle is better than the circle if the dimension is restricted. If the sharpness of the slice level is more important than the light quantity, the rectangle whose long side is set vertically is recommendable. The width of a rectangle depends on the required light quantity etc. and so can not be determined uniquely.

The number of channels is determined as follows. First the stroke width of characters is examined for the characters shown in Fig. 2.7,which shows the stroke width is 1/9--1/10 of the character height. So the resolution in the vertical direction must be at least 10. If the slice level of the input signal is set 50% or more of the maximum signal level, more than one cross section of input part must fall in a stroke width. However more than two cross sections for a stroke width is not needed because the characteristic feature logics become very complicated with redundant channels.

From these discussions 16 channels are recommended for

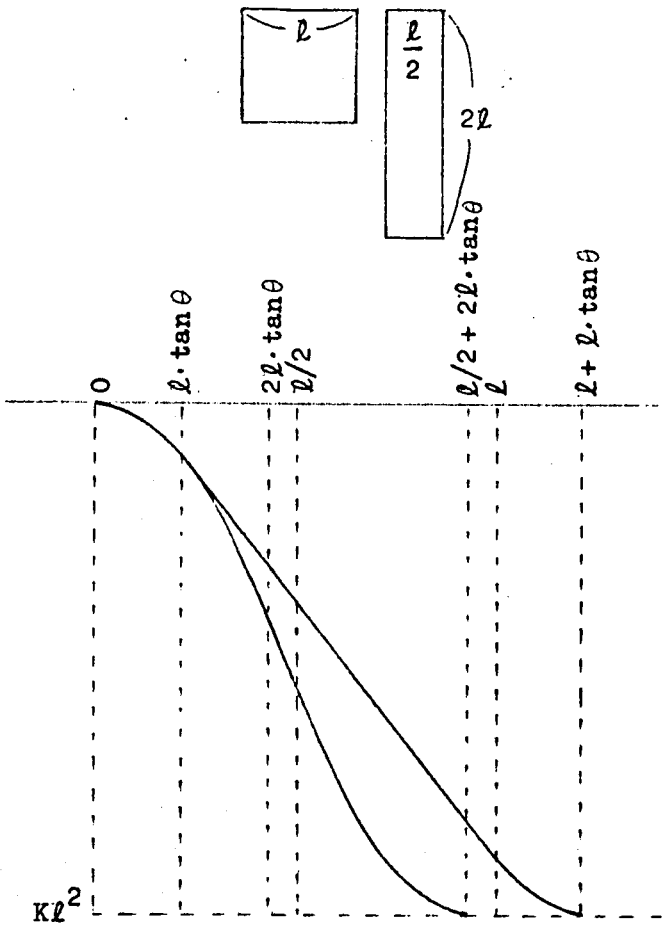


Fig. 2.31 Light quantity curves when a vertical line with a slight inclination (θ) comes in to the above cross sections.

the height of a character, in which case at least one channel will fall in the width of a stroke. Considering that the mechanics of line feed operation is not so precise, another 6 channels are added. The width of a channel corresponds to about 0.18 mm on the original paper, so that 6 channels may not be enough for more than 30 line feeds. But by the experimental machine the deviations of 4 channels was observed in bad conditions when the line feed operation was done from top to bottom of a paper.

2.3.2 Base Line Detection and Channel Exchange

One of the features of this machine is that many lines of characters in a paper are read continuously and in high speed. The mechanics of the line feed of an electric typewriter is not so precise as to have no deviation, so that certain amount of error accumulation must be allowed when more than thirty lines are feeded in a paper. To save this deviation 22 input channels are provided for, in which 16 channels are for the input characters and another 6 channels are for the deviation of the line feed. This machine determines the position of a line as a whole, not of each character, which will be mentioned in the next section. The block diagram of Fig. 2.32 shows how the channels are determined where a line of characters exists.

First the CR action is taken. The output signal of each channel during the CR action is accumulated in a condenser of pulse accumulation circuit. The input to this circuit is the pulse waveform from the input amplifier. At the end of

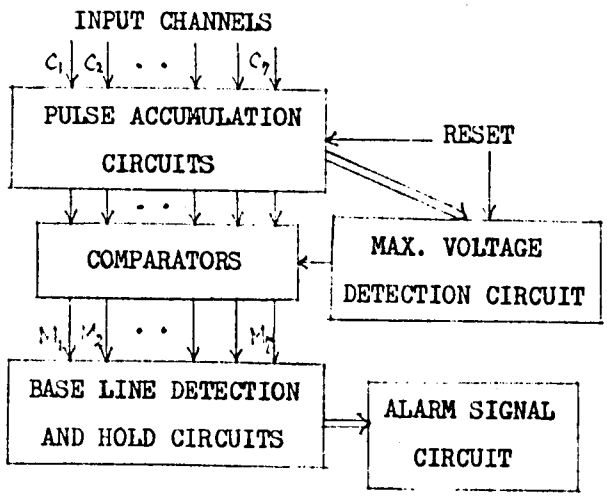


Fig. 2.32 Base line detection and hold circuits which select proper 16 channels out of 22. Actual circuits are shown in Fig. 2.46.

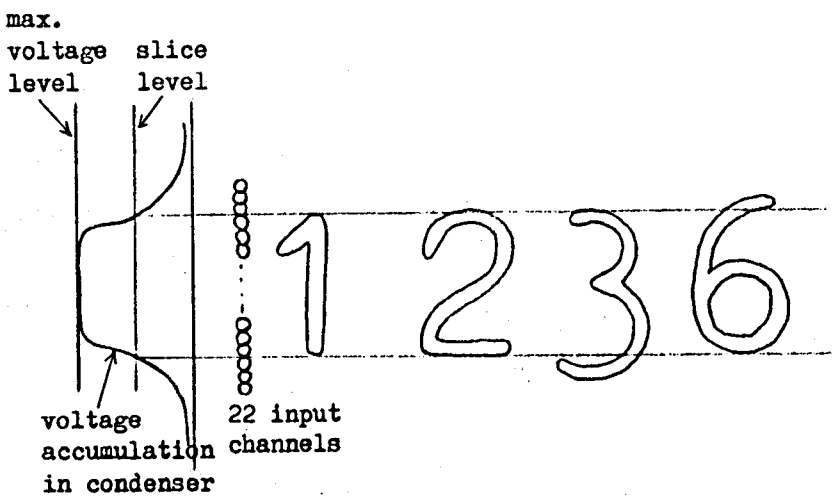


Fig. 2.33 Schematic representation of voltage accumulation in condensers of 22 channels.

the CR operation the OR logic of all the accumulated voltages are taken. The output of the OR gate is the maximum voltage level of all. Then each accumulated voltage is sliced in the comparator by a certain per cent of the maximum voltage level, which is schematically shown in Fig. 2.33.

The output "1" shows there is a portion of a character on that channel. The boundary channel of the output 1 and 0 is detected by simple combinatory logic circuit. For this purpose there is no need to attach the pulse accumulation circuit to all the channels. When 16 channels are to be selected from 22, upper or lower 7 channels are enough. Fig. 2.34 shows the circuit which detects the top channel of a character line by the pulse accumulation circuits attached to the upper 7 channels. This is called the base line detection circuit. The determination of the base line is done at the end of the CR action and just before the TAB action begins. Therefore the AND circuit is gated by the 200 milli-second output of one-shot multivibrator at the same time the TAB solenoid is activated. The detected base line is maintained in the hold circuit by the output of TAB contact during the TAB operation is being performed. The logic functions which determine the base line are,

$$B_1 = \overline{M}_1 M_2 M_3 M_4$$

$$B_2 = \overline{M}_1 \overline{M}_2 M_3 M_4$$

$$B_3 = \overline{M}_2 \overline{M}_3 M_4 M_5$$

$$B_4 = \overline{M}_3 \overline{M}_4 M_5 M_6$$

$$B_5 = \overline{M}_4 \overline{M}_5 M_6 M_7$$

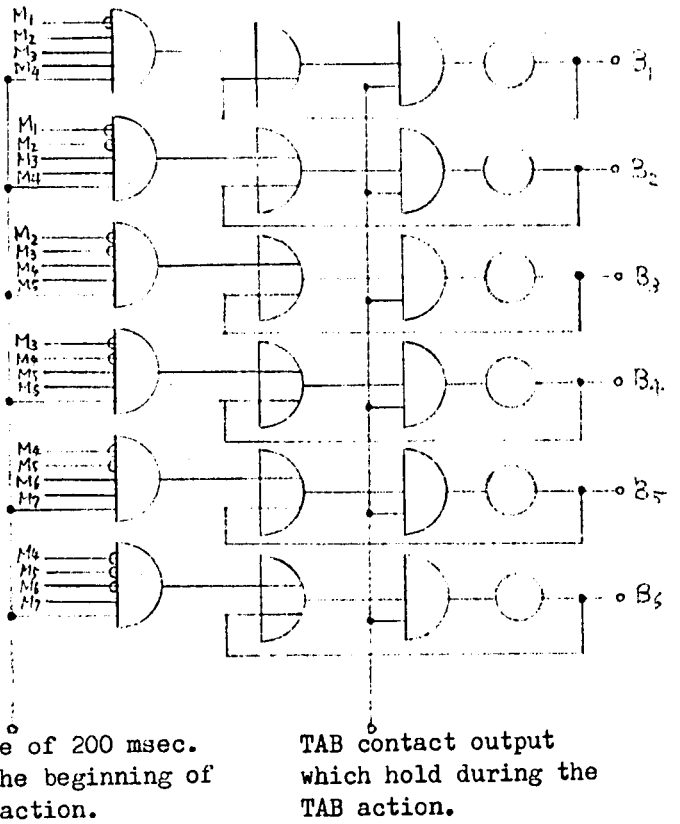


Fig. 2.34 Detection and hold circuit of the top channel of a line.

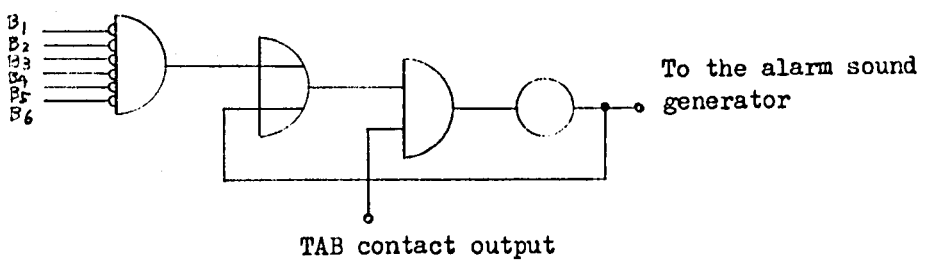


Fig. 2.35 The logic circuit for the alarm which is to be produced when the input pattern does not fall within the input channels.

$$B_6 = \overline{M}_4\ \overline{M}_5\ \overline{M}_6\ M_7$$

All these are mutually independent so that there is only one output from the circuit. If there is no output from any of B_1--B_6, this means the input pattern does not fall in the input channels, which should be indicated in some way. By this machine this is indicated by the alarm sound from the speaker. The logic circuit for this alarm signal is shown in Fig. 2.35.

When the position of a line of characters is known thus by the output of base line detection circuit, the selection of proper 16 channels from 22 can be easily achieved by the combinatory logical circuit. This is shown in Fig. 2.36 and the connection of each terminal is shown in Table 2.4.

2.3.3 Channel Exchange for Each Character

The channel exchange for a line mentioned above works well when each line is set horizontally. But when a line is inclined or the paper setting is not horizontal this method can not be applied. There may be some improvement to this, but the channel exchange for each character is to be done to avoid this kind of trouble. The following is a solution, although it is not implemented actually. This uses the delay lines as shown in Fig. 2.37. This delay line is provided for each input channel and the delay time is adjusted just the time a character passes under the input part. The input channels are also connected to the base line detection circuits which

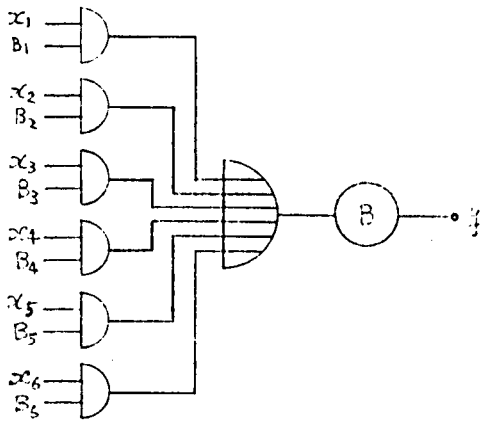


Fig. 2.36 Channel selection logics. x_i is one of 22 channels, B_i is the result of base line detection logics, and y is one of 16 channels. The connection is shown in Table 2.4.

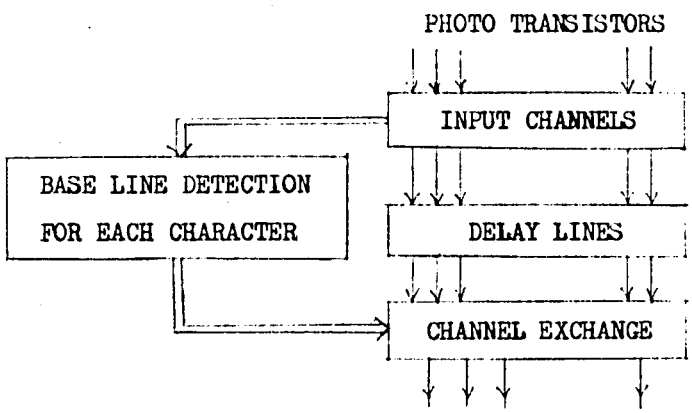


Fig. 2.37 Base line detection for each character.

Table 2.4 Channel selection logics.

y	x_1	x_2	x_3	x_4	x_5	x_6
A	C_2	C_3	C_4	C_5	C_6	C_7
B	C_3	C_4	C_5	C_6	C_7	C_8
C	C_4	C_5	C_6	C_7	C_8	C_9
D	C_5	C_6	C_7	C_8	C_9	C_{10}
E	C_6	C_7	C_8	C_9	C_{10}	C_{11}
F	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
G	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}
H	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}
h	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}
g	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}
f	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}
e	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}
d	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}
c	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}
b	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}	C_{21}
a	C_{17}	C_{18}	C_{19}	C_{20}	C_{21}	C_{22}

are similar to these mentioned above. This base line detection circuits accumulate the signal pulses in condenser for each character and determine at the end of the passage of a character the position where it is located. The information of the base line thus determined is used for the channel exchange logics which are connected to the output of the delay line.

Thus the input pattern appears at the outputs of the delay lines just after the channel exchange is completed. The condenser charge is reset to the initial state when a character has passed and the base line is determined.

Since the character speed of the machine is at now 7--8 milli-second per character, and the input signal has about 10 kc band width, the electro magnetic delay lines are not suitable. The high speed magnetic drum is recommended. This is to be used without clock cycle so that the write, read and erase heads are necessary as shown in Fig. 2.38.

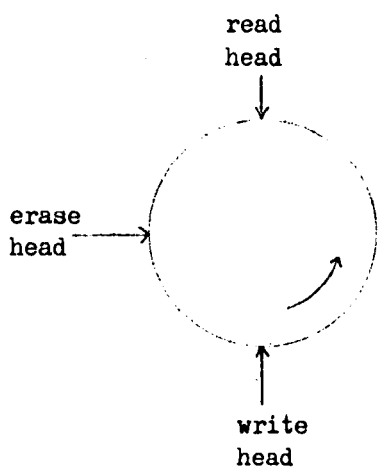


Fig. 2.38 Magnetic drum delay line.

2.4 Other Auxiliary Devices

2.4.1 Word and Character Counter

In the handling of many kinds of documents, it is very convenient to introduce format symbols. Here two special symbols < and > are introduced to bracket characters to form a word.

For example

<1234> <5678> <90123456>

are the three words, each may have certain meaning such as the first is a region number, the second is a block number, etc.

The word counter counts the appearance of special symbol < , and if the counter is set to 3, then the recognition outputs are only for the characters contained in the third word from the left of a line. The character counter counts the number of characters from the special symbol <, so that if the word counter is set to 3 and the character counter is set to 2, then only the second character of the third word is obtained as output. The counter setting can be done on the control panel by switches. If the word counter setting is 0 then only the character counter works from the left of a line and ignores the symbol <. If the character counter is set to 0, then only the word counter works and all the characters of a specified word are read.

If both counters are set to zero, then all the characters of a line are read. Each counter can count up to 16 with four flip-flops in series.

The trigger inputs to the counters come from the OR gate

of all the recognition outputs for the character counter, and from the output of special symbol < for the word counter.

2.4.2 Noise Elimination Logic

The recognition output is sent to the coder of 6 bits during the 600 μsec. directly after the condition that a character has completely passed through the input part, i.e.

$$\overline{Y} = \overline{A + B + C + D + E + F + G + H + h + g + f + e + d + c + b + a}$$

$$= \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}\,\overline{F}\,\overline{G}\,\overline{H}\,\overline{h}\,\overline{g}\,\overline{f}\,\overline{e}\,\overline{d}\,\overline{c}\,\overline{b}\,\overline{a} = 1$$

is satisfied. Therefore if there is no output during this recognition period, it implies that the input pattern can not be recognized and the reject signal is produced. If there are dots or some noises such as shown in Fig. 2.39, the recognition periods are set up at the end of these noises by the condition $\overline{Y} = 1$, i.e. these noises are regarded as input patterns so that some recognitions are to be produced. However as these are actually not characters to be recognized but only noises, the outputs (even if they are reject signals) are to be suppressed.

The distinction of the characters and noises is done in the following way. That is, if an input pattern is a character the following conditions are to be satisfied during the passage of a character.

$$W_U = B + C + D = 1$$

$$W_M = G + H + h = 1$$

$$W_L = d + c + b = 1$$

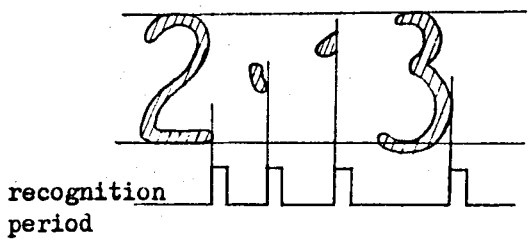


Fig. 2.39 Recognition period for characters and noises.

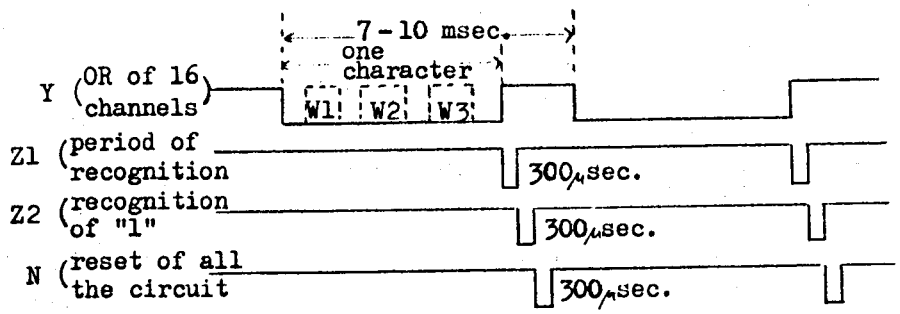


Fig. 2.40 Recognition timing of a character.

The outputs of these logics W_U, W_M, and W_L are memorized in circuits and at the end of the input, the condition

$$W_U \cdot W_M \cdot W_L = 1$$

is satisfied for a character. Otherwise the input is regarded as noise and the recognition output is suppressed. The above condition means that a character has output on the upper, middle and lower zones of channels at sometime during the passage of the character.

2.4.3 Timing Control of the Machine

The timing of the whole machine operation is shown in Fig. 2.25, and Fig. 2.40. Referring to Fig. 2.25 and Fig. 2.26, first the CR solenoid is activated and the CR movement begins. During this CR movement there is output on the CR contact line and the base line detector condensers accumulate the pattern signals. At the end of the CR movement the CR contact opens and the one-shot multivibrator of 100 msec. is triggered. At the end of the output of this one-shot multivibrator, other one-shot multivibrators of 100 msec. and 200 msec. are triggered. This 100 msec. pulse output activates the TAB solenoid and the TAB movement begins. During this TAB movement the TAB contact has output. The region of this horizontal movement can be adjusted wide or narrow by the TAB setting. The pulse output of 200 msec. is used for the gating of the base line detection logics of Fig. 2.34.

At the end of the TAB movement the TAB contact opens and one-shot multivibrator of 100 msec. is triggered. After this

the CR solenoid is again activated by another 100 msec. multivibrator and the same operation is repeated. The line feed is done at the beginning of the CR movement.

Fig. 2.40 shows the timing for the recognition of each character. The logical function Y = A + B + ···+ G + H + h + g + ···+ b + a has output during the passage of a character.

At the end of the Y pulse a one-shot multivibrator is triggered, which produces the recognition pulse Z1 of 300 μsec. for usual characters. Next in sequence the second multivibrator Z2 of 300 μsec. is triggered for the recognition of special character 1. Finally the third multivibrator N of 300 μsec. is triggered for the reset of all the circuits of the recognition to the initial state.

2.4.4 Monitor Display

A pattern on a paper and a signal pattern received in a device via a proper photo electric conversion are different. Several factors of the input device are present. The recognition device works on this received signal pattern, so that it is very important to know how the original pattern is received in electric signal. This makes exact the design of pattern recognition machine, as well as its maintenance. The received signal pattern of this machine is continuous in time axis although in the vertical direction it is digitized in 16 channels. So that this pattern signal can not be displayed in this signal form on the cathode ray tube.

Therefore the signal is sampled in time axis and this

sampled pattern is displayed on the cathode ray tube. The input signal is obtained either from the 16 outputs of channel exchange or from the 22 outputs of input amplifiers. The selection is done by a switch. The fundamental sampling frequency is given from outside by a sine wave oscillator, whose upper limit is 70 kc/sec.

Y axis of the cathode ray tube is given a saw tooth wave which repeats every 32 counts of the fundamental sampling frequency. The pattern signals on 16 channels are sequentially sampled from the first channel to the 16th or to the 22nd by the AND gate with counter outputs from 0 to 31. These sampled signals in pulse train are inserted to the Z axis of the cathode ray tube as brightness control and the sampled pattern appears on the cathode ray tube. Some sampled patterns displayed are shown in Fig. 2.41.

By watching the sampled pattern display the slice levels of the input amplifiers and thus the width of strokes of characters can be adjusted. When the display is done by 16 channels, how the base line detection logics are working can be seen.

2.4.5 Output

The recognition output can be converted in any form according to the demand of the system. The character recognition machine developed here is for experiments, so that it is required to see if the recognition outputs are correct or not. Therefore the recognition outputs are stored in the

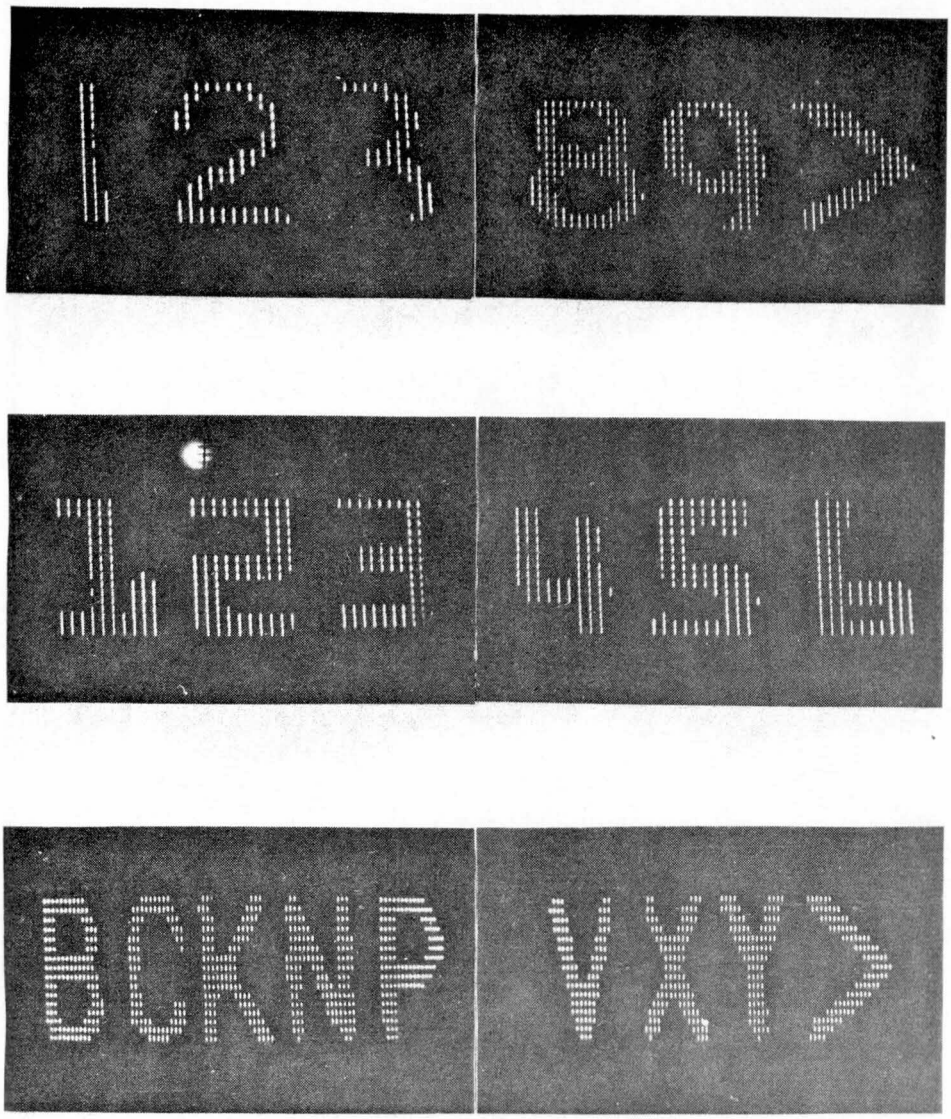


Fig. 2.41 Monitor display of received patterns.

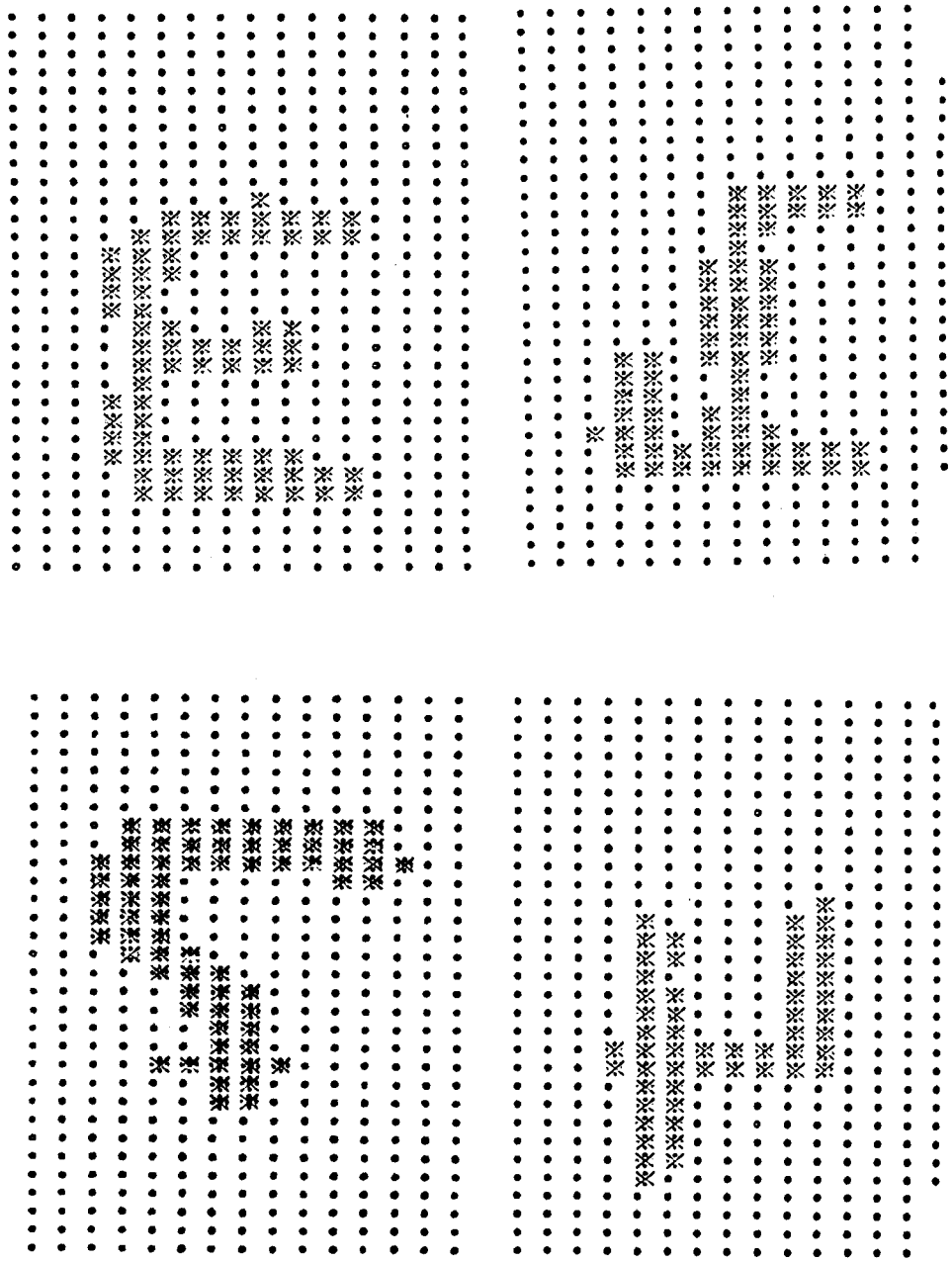


Fig. 2.42 Sample patterns printed out.
Left and right are reversed by printing sequence.

magnetic core memory and then printed out to the typewriter. This buffer memory is required because the reading speed of characters is 100 characters/sec, but the printing speed is 10 characters/sec.

The recognition results are coded in 6 bits parallel code and are stored in buffer memory in ascending order. When a line of characters are recognized and stored in buffer memory, the carriage return code is inserted for the results to be printed out in the same form as the original.

Properties of the magnetic core memory:

read, write: 1.5 μsec.

1 word : 8 bits

memory size: 1024 words

For the details it is hoped to refer to the bibliography (20). The sampled pattern for the display on the cathode ray tube can also be stored in this core memory and printed out on a paper, which is very convenient to see. Few samples printed out is shown in Fig. 2.42. At the same time this sampled pattern can be punched out to a paper tape which is used for the input to the digital computers installed in Kyoto University.

2.5 Circuits of the Machine

2.5.1 Introduction

The pattern recognition machine constructed is composed of input typewriter, photo electric conversion system, base line detection and channel exchange circuits, characteristic feature logics, recognition logics, output buffer register,

output typewriter and some other circuits. The mechanical parts are the input and output typewriters and all others are electronic circuits.

The overall view of this system is shown in Fig. 2.4. The main part containing electronic circuits is 110 cm × 60 cm × 130 cm, which is made to have much space for the experiments and alterations to be done easily. Transistors and diodes are used for all the circuits, every unit of which is made in a package. The main specifications of the machine is as follows.

(1) Characters accepted : NEAC typer 0-- 9, A, B, C, K, N, P, R, S, U, V, X, Y, < , >
OCR-A (ISO) 0--9,

(2) Number of recognition logics :
symmetric character, 4
unsymmetric character, 7
built in, 3

(3) Number of transistors : about 1200
" diodes : about 1500
(the core memory is not included)

(4) Recognition speed : about 100 characters /sec.
(excluding the time for carriage return)

All the circuits work with -6V signal pulse. Since the input signal is not so fast, extremely high speed operation is not required for the circuits. The circuits work up to 80 kc/sec, and the actual working frequency may be 30--50 kc/sec. In the following some of the representative circuits are ex-

plained.

2.5.2 Input Amplifier

By this machine the most important and difficult problem was to amplify a very small input signal. The intensity of the reflected light from a document paper is so weak and the attachment of a small glass tube to the head of photo transistor is not so precise that the output current of the photo transistor is very small and noisy. The DC amplifier was hoped, but considering the DC level shift the circuit is made as AC amplifier which is shown in Fig. 2.43. The frequency characteristic of this circuit covers the range 50c/sec--20kc/sec, which latter is the upper limit of the photo transistor. This circuit, however, has no function of temperature compensation, so that by the temperature shift of about 7°C, the width of a received pattern changes considerably. The circuit shown in Fig. 2.43 has a comparator which slices input signal by some DC level, which can be adjusted manually for all the channels simultaneously. The circuit at the input side which is composed of transistors Q_3 and Q_4 works as the level adjustment of the input pulse trains.

2.5.3 Logic Circuit

The circuit has no clock pulse, but works asynchronously only when a pulse comes in to the circuit. However the response of all the circuits is not the same, and very narrow noisy pulses are better to be absorbed, so that the Schmidt

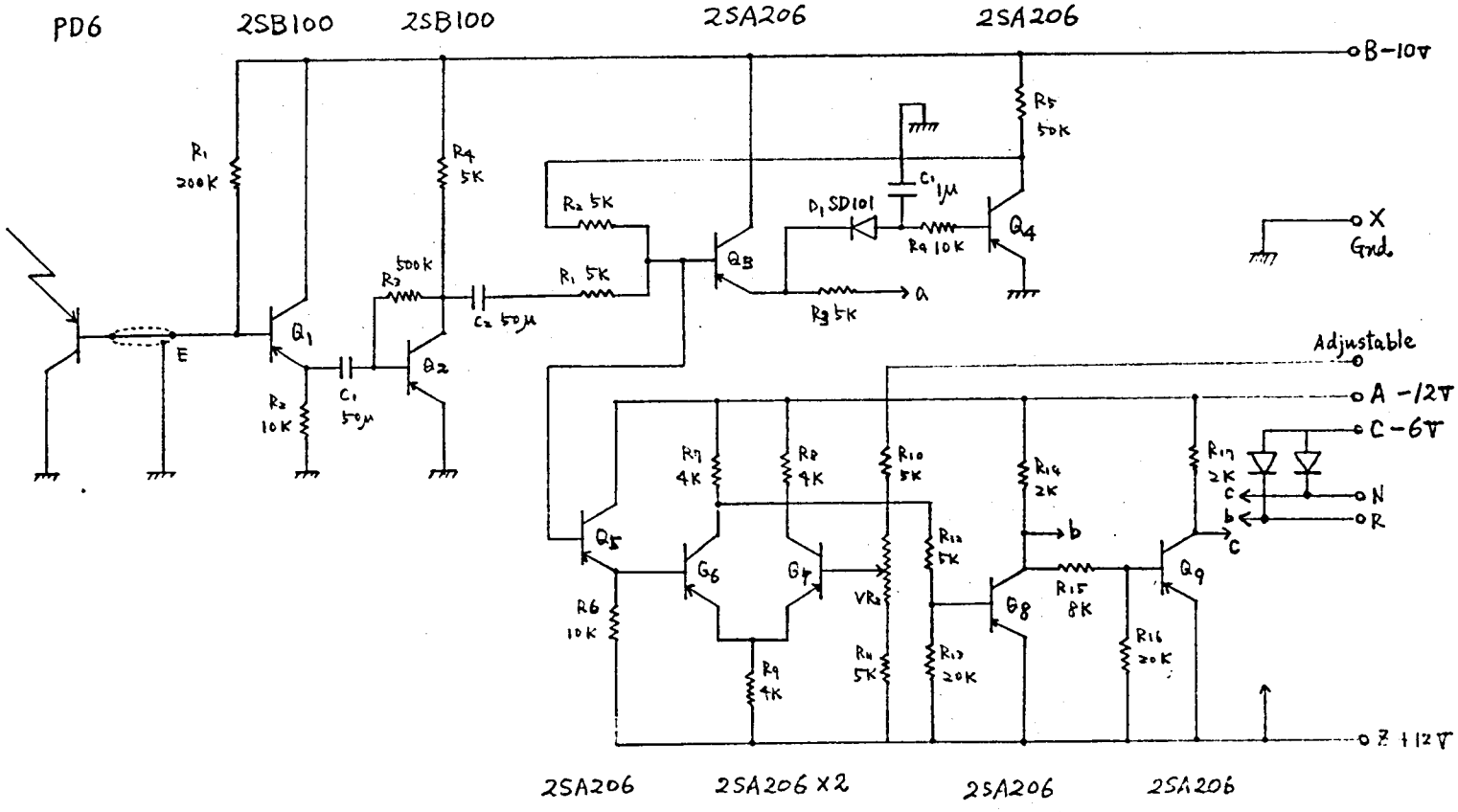


Fig. 2.43 Input amplifier with slicer.

circuit as shown in Fig. 2.44 is used with the CR integrator in the input side. This condenser absorbs noisy pulses and also has the effect to delay the input pulse. The recognition circuit is composed of this circuit as shown in Fig. 2.45. The feed back line from the output to the input constitutes one bit memory circuit.

2.5.4 Base Line Detection Circuit

The voltage accumulation circuit for the detection of base line is shown in Fig. 2.46. The input is a signal pulse from an input amplifier. The condenser is connected to the line H and accumulates the voltage by the backward high resistance of two diodes and by the high grid resistance of triode 20-DL5. Five kinds of condensers 10μF, 5μF, 2μF, 1μF, 0.5μF can be connected to the line H by a selection switch. This is changed as the number of characters printed in a line differs extremely. The diode gate of Fig. 2.46b is an OR circuit of all the output voltages of the accumulation circuits attached to upper seven input channels, and the condenser of the succeeding circuit holds the highest accumulated voltage. Certain per cent of this maximum voltage is used as a reference of comparator of each base line detection circuit. The line P is provided for the reset of the condenser voltage which occurs at the end of TAB operation.

2.6 Recognition Results and the Factors Affecting the Recognition Rate

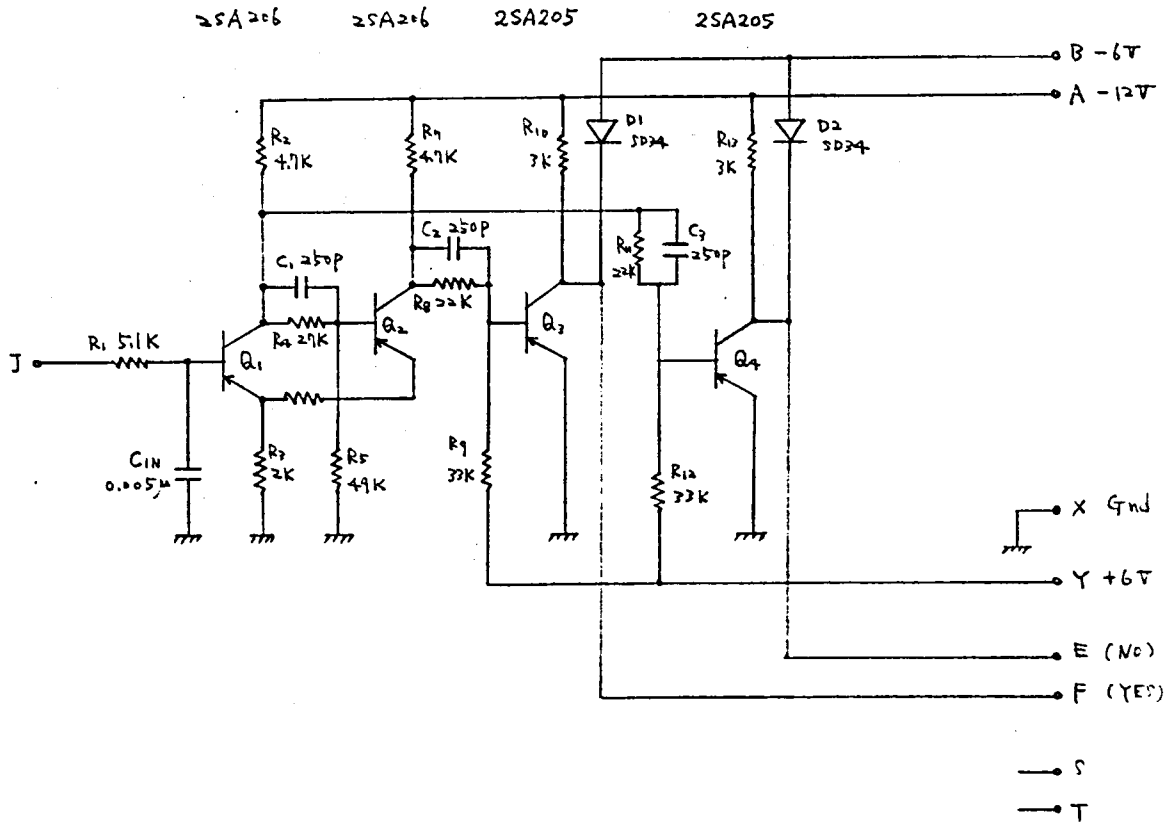


Fig. 2.44 Schmidt circuit which regenerates input pulse. Three units are in a package.

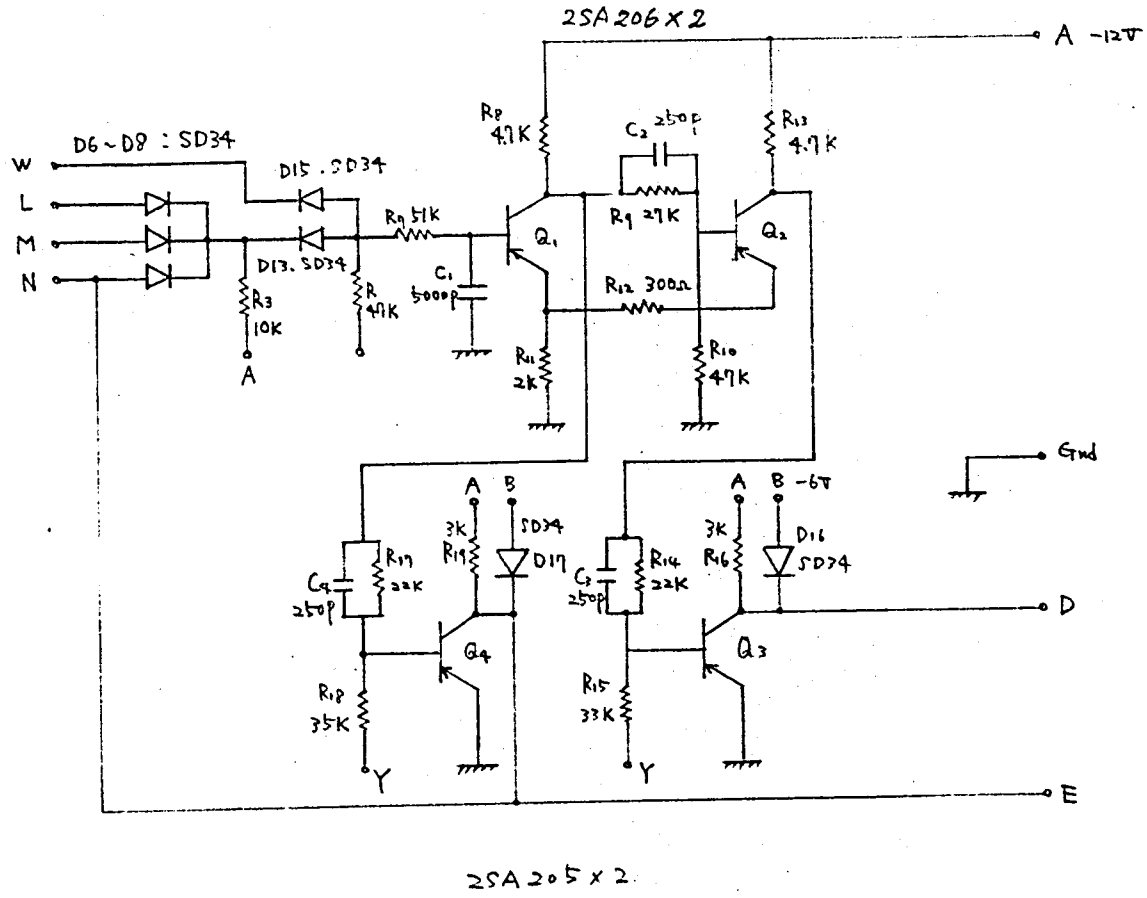


Fig. 2.45 Recognition circuit with one bit memory.

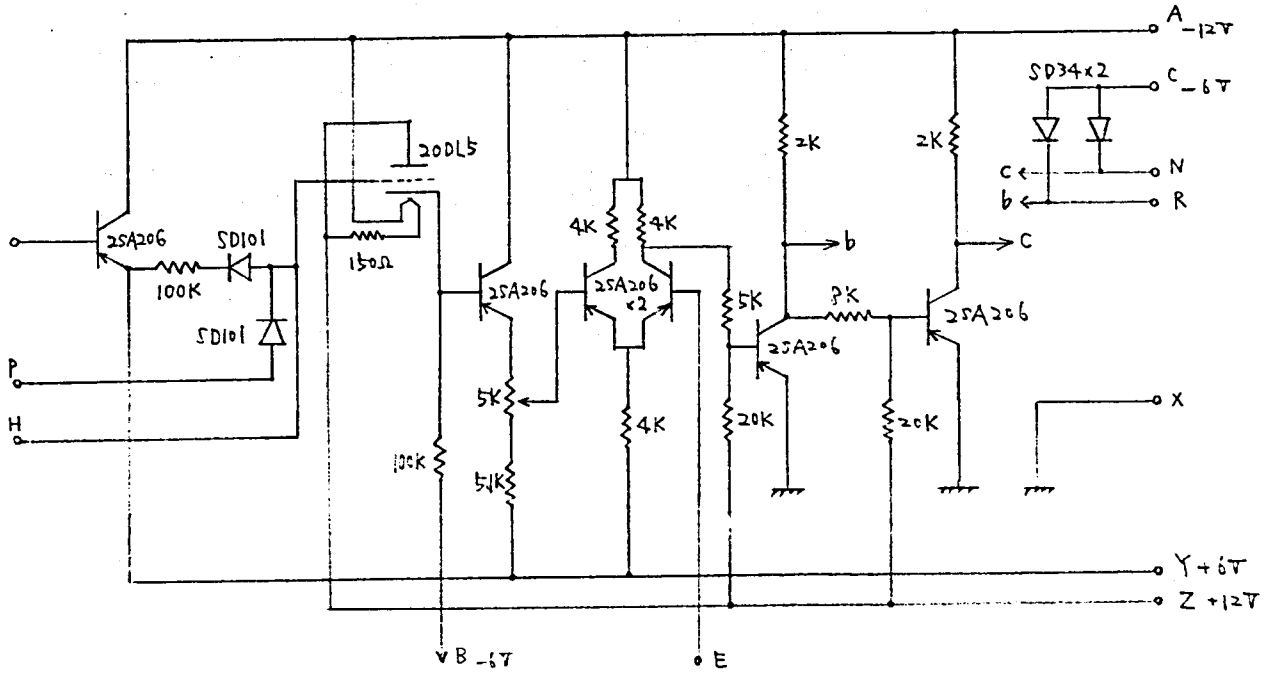


Fig. 2.46a Voltage accumulation circuit for the detection of base line. Five kinds of condensers are connected to H by a selection switch. P is the line for the reset.

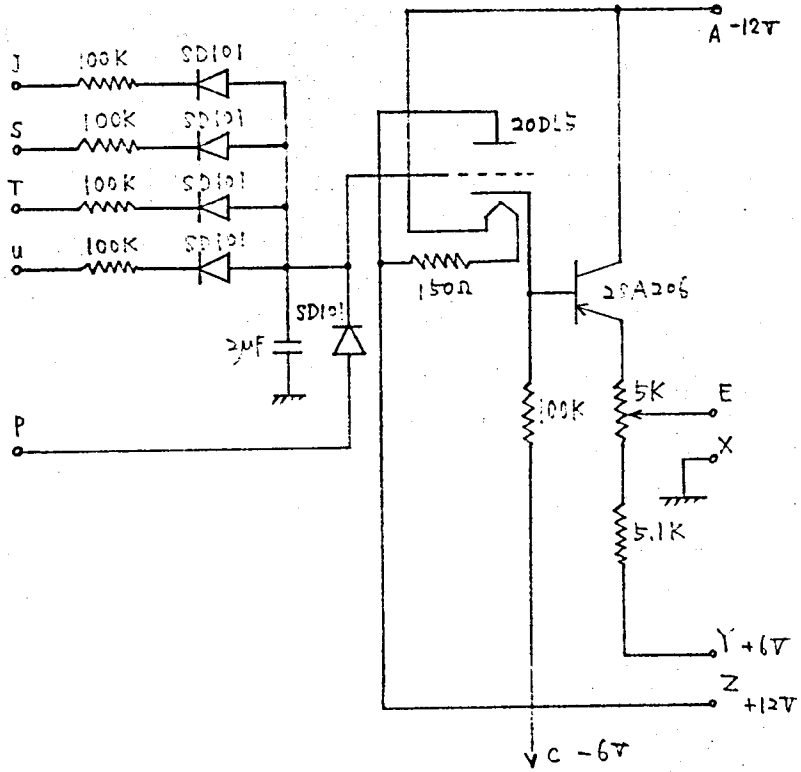


Fig. 2.46b Maximum voltage detection circuit. Output E is connected to the comparator of Fig 2.46a.

2.6.1 Recognition Results

The experiment on this system was done in the following two steps. The first was to testify the validity of the recognition principle, and the second was to know the accuracy of the recognition to various kinds of print quality.

Some of the characteristic feature logics were amended because the received signal patterns were noisy and the vertical shift from the normal position was more than expected. Next there were some cases where the unexpected characteristic feature logics appeared in unexpected places. For example the third characteristic feature logic W3 for a character appeared very often before the first or second characteristic feature logics. By the recognition principle mentioned in 2.2.4 this inhibits the recognition output. One of the solution, when the pulses of such unexpected characteristic feature logics were very narrow, was to absorb the pulses by inserting comparatively large capacitor (0.05μF to 0.4μF) to the input of logic circuit shown in Fig. 2.44. If the unexpected pulse is not so narrow as to be absorbed without any effect to the

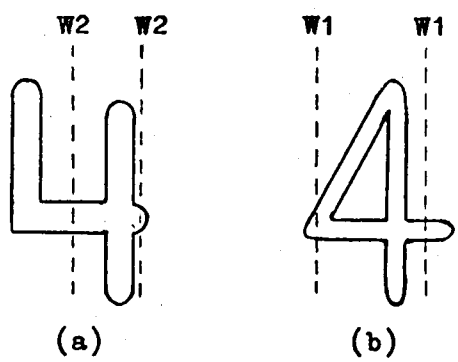


Fig. 2.47 Example of characters which need special considerations.

normal pulse, the recognition circuit was changed to that explained in 2.2.3. For example the recognition of character "4" is by this circuit. This situation is shown in Fig. 2.47. In Fig. 2.47a W2 appears again after W3, and in Fig. 2.47b. W1 is possible to appear after W3. The confusion of 2 and 8 of Fig. 2.18 was avoided by the insertion of capacitor to the third characteristic feature logics of 2.

In this way fine adjustment was done to the characteristic feature logics and to the recognition circuits whenever an error appeared. The sample patterns were printed by an electric typewriter but many factors had effects on the pattern quality, such as type ribbons, typing pressure and so on. The experiment was done by samples as shown in Fig. 2.48. The average error rate was 0.1% and the average rejection rate was 0.2% for the samples of more than 10,000 numerals of ordinary type font, which were typed by one time carbon ribbon. For the alphabetic characters the error rate, the rejection rate inclusive, was about 0.5%.

The error may be divided in two kinds. One is an input character can not be recognized, and the other is an input character is recognized as a different character or two recognition outputs are obtained simultaneously. For the numerals, examples of the former case were 2, 4, 5, and 8. The reason why 2 could not be recognized is that the third characteristic feature logic appeared before the first one, and the inhibition logic was activated. A latter case is that a character 6 was recognized as both 6 and 5. In this case 5

Fig. 2.48 Examples of sample patterns.

was not recognized as 6, so that if 6 and 5 are simultaneously recognized, the output of 5 may be suppressed. Another case was a character 1 of OCR-A type font was recognized as 6. This is because there was a noise as shown in Fig. 2.49.

There was a special cause of error for OCR-A font. For example for the character "2" the end of the second characteristic feature logic was later than the beginning of the third one as shown in Fig. 2.50, which activated the inhibition logic. This is that the circuit delay time of the former is longer than that of the latter, so that this cause was avoided by the adjustment of the delay time of the circuits. For the ordinary characters there are some transient interval from a characteristic feature logic to another, in which case this kind of error does not occur. The error rate of the characters typed by ordinary cloth ribbon was about twice of the error rate by one time carbon ribbon. When the typing pressure is not suitable the recognition rate decreases considerably.

2.6.2 Factors Affecting the Recognition Rate

There are many factors affecting the recognition rate. Some are specific to this machine and others are not specific but are generally applied to all the optical character recognition machines.

(1) Character shapes

When there are many typewriters, even if they have all the same type faces, there exist little differences. For the OCR-A of ISO, the type faces casted were a little

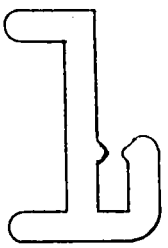


Fig. 2.49 Character 1 which is misrecognized as character 6 of OCR-A.

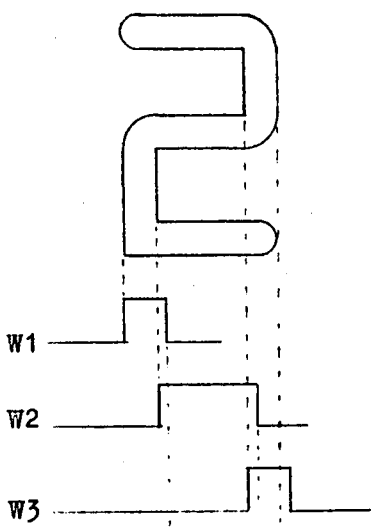


Fig. 2.50 Overlap of characteristic feature logics.

different from the character shapes of original design with which the logical design was made. Important points in casting seem to be the edge of a character shape, the curvature of the type face and the hardness of the material.

(2) Character size

The magnification of an optical image to the prescribed size is to be correct. The focus should, of course, be sharp. Since the magnification rate is about 3, and the distance between a paper surface and the object lens is not long, even a slight floating-up of a paper from the platen during the line feed has a serious effect on the focus plane.

(3) Light

The light intensity is to be enough for the sensitivity of the photo transistor. DC is to be used for the lamp. The light intensity is varied by the DC voltage supply and thus the width of strokes of input characters can be changed for a certain amount. However this is closely related to the slice level of input amplifier.

(4) Paper

There could not be found a severe specification on the quality of paper. However the paper opacity must be adequate, so that the background has no effect to the reflected light. Such kind of papers on which a hollow is made by typing is to be avoided, because there may occur certain irregularity of reflections. Of course the

dirt on paper is not good. However if the dirt is very small not to be recognized at a glance, its effect on the recognition is too small to be expected.

(5) Paper setting

The paper must be set strictly parallel to the scanning direction. The line skew of two channels between both ends, which is about 0.36 mm, will cause errors. This is one of the weakest point of the machine.

(6) Character alignment

Character alignment in a line must be as exact as possible. The misalignment is allowed only one channel up or down. This is from the design of characteristic feature logics. The effect of a character skew was not examined extensively, but among the characters of OCR-A which are attached to the typewriter, there were few whose skew exceeded 3°. These were of course recognized correctly.

(7) Print quality

By the print quality the following factors are at least to be considered.

(i) High contrast to the background document.

This was not measured by the experiment. But as far as certain contrast is held, colored background may be allowed. Actually a pale blue paper was tested with good result. The contrast depends largely on the pressure of printing and ink ribbons.

(ii) Stroke width should be held as close as possible

to the nominal.

If the stroke widths are very narrow, the received pulses may be absorbed as noise pulses. If it is too broad, the lines closely positioned may be regarded as connected. This also has close relation to the type ribbon and pressure of typing.

(iii) There should be no extraneous ink or no voids. Extraneous ink usually appears when cloth type ribbon is used and the typing pressure is large. Large voids often occur within the stroke outline, when a one time carbon ribbon is used.

(iv) Stroke edges are to be as sharp as possible. This depends largely on the type ribbon and the quality of type face.

(v) Typing pressure must be equal within a type face. If a type face is not parallel to the paper when typing, there occurs irregularity within a character. An example is shown in Fig. 2.51. This kind of typing is very bad for the recognition.

Few samples of the actual typewritten characters are shown in Fig. 2.52. Characters typed by one time carbon ribbon and by cloth ribbon are drastically different.

(8) Type ribbon and typing pressure.

All the factors listed in (7) are influenced largely by the quality of type ribbon. There are so many kinds of type ribbons and the differences among them can not

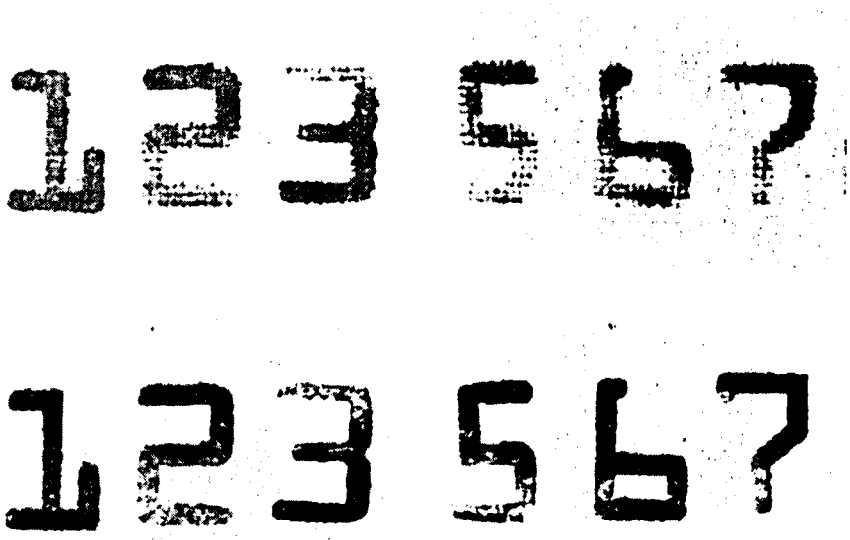


Fig. 2.51 Examples of unequal pressure on top to bottom.

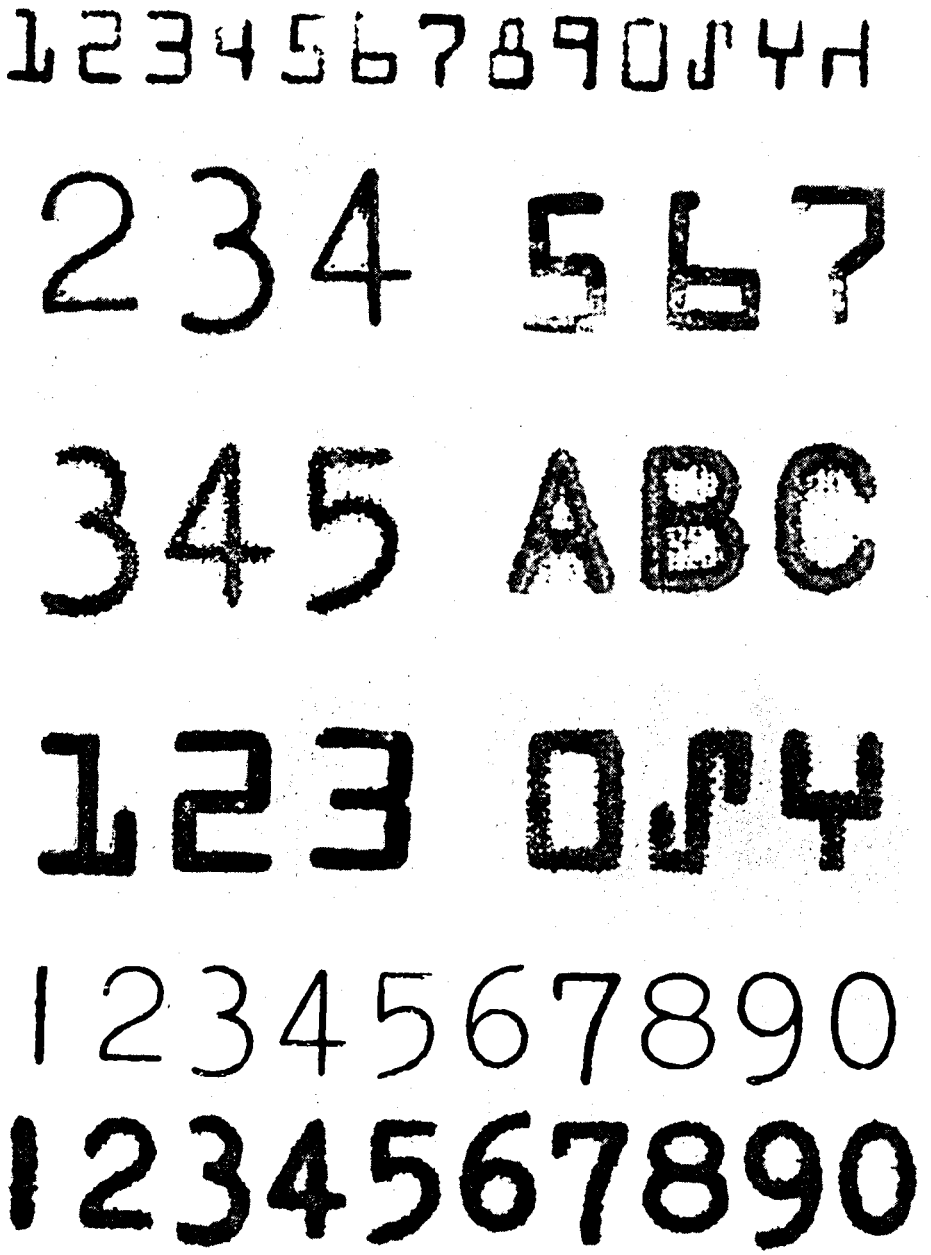


Fig. 2.52 Several kinds of print quality.

be measured explicitly, so that the problem becomes more difficult to treat. However the difference between the one time carbon ribbons and the ordinary ribbons is clear. Few examples of typed characters by different type ribbons are shown in Fig. 2.52. By one time carbon ribbon the stroke edge is usually clear but often there occur large voids. Also extraneous ink is apt to appear at the corners.

By ordinary cloth type ribbons, the stroke edge is irregular and the fine network of the cloth becomes the cause of extraneous ink when the typing pressure is strong.

(9) Circuit.

The temperature compensation is not done for the input amplifier, so that the temperature effect is large. The optimum temperature is between 20°C--25°C, although the optimum range can be varied by the readjustment of the slice level, lamp intensity, and a supply voltage to the input circuit.

CHAPTER 3

VARIATIONS OF THE PRINCIPLE

3.1 Separation of Upper and Lower Halves of Input Channels

The character recognition machine constructed here by the principle mentioned in Chapter 2 needs three characteristic feature logics for each character. So thirty logic functions are needed for the recognition of ten numerals. As there are some functions which are just the same, the number is less than thirty. Actually the number of logic functions is twenty for the numerals of ordinary type font, and seventeen for the numerals of OCR-A font.

An idea for the decrease of the number of logic functions is to separate the characters in several parts, each part of which is the same for many characters. The research work done by Mr. Masami Tanaka[11] and us[12] on hand written characters was from this standpoint.

The object of recognition was handwritten numerals, zero to nine, and the channels were seven. The characters were separated in upper and lower halves.

The subpatterns were grouped in ten categories, which are shown in Table. 3.1. For the detection of each subpatterns the three characteristic logic functions are to be provided.

The recognition of a character is done by the combination of two proper outputs from the categories of upper and lower halves of a character. The block diagram of this machine is shown in Fig. 3.1. The logic functions are set up among the

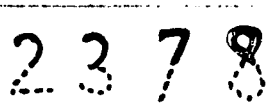
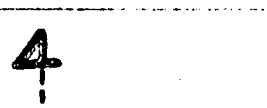
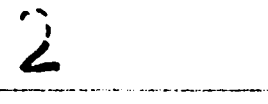
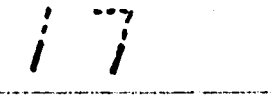
Upper subpatterns	1	
	2	
	3	
	4	
Lower subpatterns	5	
	6	
	7	
	8	
	9	
	10	

Table 3.1 Upper and lower subpatterns of 10 numerals.

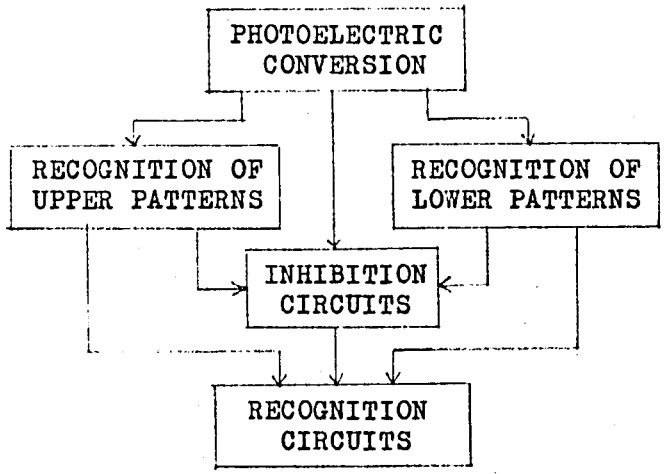


Fig. 3.1 Block diagram of character recognition machine which recognizes upper and lower patterns independently.

upper four channels and the lower four channels, using the central channel in common.

From this table of subpatterns it may be seen that the grouping is not suitable, and also the ten groups are not enough. For example 8 in the group 2, or 4 and 9 of group 8 are not suitable. By this method the characteristic feature logics are 30. A deficiency of this method is that an upper subpattern and a lower subpattern are recognized independently, and there is no consideration about the mutual relation between upper and lower parts, except the central channel is in common use. When the characteristic feature logics must satisfy the characters of one channel up or down, subdivision of a character is not good, because the number of channels becomes few, and the logic functions are very difficult to be determined for such demand.

Thus this method is inferior to that explained in Chapter 2, in that this does not economize the circuits and the recognition rate does not improve. However this was the first experiment in character recognition which used the asynchronous principle, for which it has a value.

3.2 Recognition Method with Two Vertical Line Inputs

3.2.1 Principle

With the principle mentioned in Chapter 2, there is a difficulty in the recognition of E, F, I etc. This comes from the recognition principle, i.e. the inability to recognize the horizontal length. A long horizontal line and a short one have no difference by the principle. To save this deficiency it might be good to introduce an absolute time interval to measure the length as illustrated in 2.2.5. This, however, limits the change of the scanning speed to a certain extent. Another method is to have auxiliary input channels which are separated from the main input channels, and by which the horizontal length is measured. This is schematically shown in Fig. 3.2. The input channel which scans the input pattern first is named I, and the other input channel which scans the input pattern later is named II. The characteristic feature logics are constructed by the combination of the logic functions set up for the channels I and II. The detection of a character is done by the sequence of appearance of these characteristic feature logic functions set up for the two channels.

An important problem by this method is to determine the

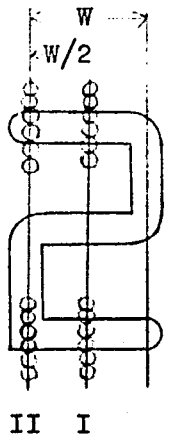


Fig. 3.2 Two vertical lines of input.

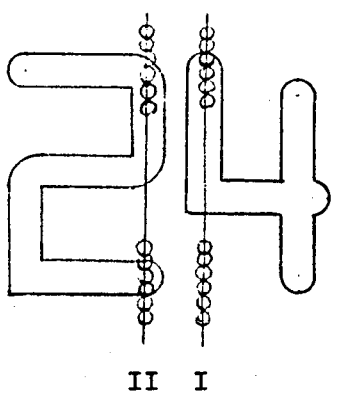


Fig. 3.3 Two input lines which step over two character.

distance of these two channels.

(1) If they are positioned too close, the distinction between noises and the portions of a pattern is difficult. The two logic functions set up for these two channels are to be completely different, which can not be realized in this case.

(2) If they are positioned too separate, some characters may fall in between these two channels, which is not preferable for this method.

The design by this principle is done for the OCR-A characters which are shown in Fig. 2.6. In this case the distance between these two is set to W/2. The 16 channels are supposed to be aligned vertically also in this case. Here also inhibition circuit is used for the suppression of unpreferable recognition output.

For example the recognition of the character "2" shown in Fig. 3.2 is done in the following way. The first vertical sections for two input parts are denoted by I and II , while the second vertical sections are by I' and II'. The logic functions provided for each vertical section are,

$$\text{I} \quad : Q4 = (A + B)\,\overline{D}\,\overline{E}\,\overline{F}\,(H + h)\,\overline{f}\,\overline{e}\,\overline{d}\,(b + a) = 1$$

$$\text{II} \quad : Q5 = (A + B)\,\overline{D}\,\overline{E}\,\overline{F}\,\overline{h}\,\overline{g}\,f\,e\,d\,c\,b = 1$$

$$\text{I}' \quad : Q6 = B\,C\,D\,E\,F\,G\,H\,\overline{f}\,\overline{e}\,\overline{d}\,(b + a) = 1$$

$$\text{II}' \quad : Q4 = (A + B)\,\overline{D}\,\overline{E}\,\overline{F}\,(H + h)\,\overline{f}\,\overline{e}\,\overline{d}\,(b + a) = 1$$

The recognition of 2 is done by the satisfaction of logic functions $Q4 \cdot Q5 = 1$ and $Q6 \cdot Q4 = 1$ sequentially in this order.

Now when these two input parts are provided, there arises

a situation where I is on the left of a character and II is on the right of the next, which is shown in Fig. 3.3. And in the cases like this, there is a possibility that a certain logical condition is satisfied and an output is produced. Therefore some suitable means are to be provided for the recognition intervals.

Characters are classified in three groups from this standpoint.

(1) Vertical line, period, comma, colon, semicolon, apostrophe

(2) equal, plus, asterisk, quotation mark, hyphen

(3) The other remaining characters.

The character widths of group 1 are narrower than these of groups 2 and 3, and also narrower than the distance between input parts I and II.

The characteristic feature logics for the characters of groups 1 and 2 are made not to be satisfied by any feature logics for the characters of group 3. (The reverse condition is not demanded.) In the other cases when some confusions arise, proper inhibition is performed.

The start and end signals for reading and recognition timing are determined in the following way.

(1) The start signal of reading.

For all the characters the start signal is given when there first appears a portion of a character under the input part II.

(2) The end signal of reading.

For the characters of group 1, the end signal of reading is given when a character has passed completely under the input part II. For the characters of groups 2 and 3, the end signal of reading is given when a character has passed completely under the input part I.

(3) The recognition timing.

For all the characters the recognition timing is provided just after the end signal of reading.

In this way the recognition timing for the characters of group 3 is earlier than the timing for the characters of groups 1 and 2, so that the recognition output from the characters of group 3 can be used as the suppression of the output from these of groups 1 and 2.

3.2.2 Characteristic Feature Logics

The principle of the determination of characteristic feature logics is the following.

(1) Even if an input pattern shifts one channel up or down, the characteristic feature logics are to be satisfied.

(2) Even if there are some extraneous noises or voids, the characteristic feature logics are to be satisfied.

(3) Inhibition logics are to be determined, by taking into consideration the inclination of an input character.

The positions where these logical conditions are to be tested are another important problem. This is done by the principle that two conditions established in the two points of

a character are different as far as possible. Here minimization of these logical conditions has not been done from the following reason. If the minimization is done too far, the logical conditions become very weak for noises and voids, which leads to the increase of error rate. If the conditions are rather strict, the rejection rate may increase, while the error rate will decrease. For the design of practical machine rejection may be preferable to error if the total rate of reject and error can not be decreased.

In Fig. 2.6 the first position for the logical condition is indicated by the symbol ".", while the second position is by the symbol "x". The right points of either "." or "x" are for the input vertical line I, and the left points are for the input vertical line II.

The characteristic feature logics established for each position of "." and "x" are listed in Table 3.2.

The pairs of characteristic feature logics which are used for the recognition of characters are listed in Table 3.3. For example 2_1 means the first position for the numeral 2, for which the logic function Q4·Q5 is provided. 2_2 means the second position and for which the logic function Q6·Q4 is provided. However it should be noted that the two Q4's here are different in actual installation, because the input parts are different. Thus the logic functions which are to be provided for the input parts I and II are as follows.

(a) The characteristic feature logics for input part I :

Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q9, Q10, Q12, Q13, Q14, Q15,

Table 3.2 Characteristic feature logics.

Q1 = B C D E F G H h g f e d c b

Q2 = (A + B) $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ $\overline{h}$ $\overline{g}$ $\overline{f}$ $\overline{e}$ $\overline{d}$ (b + a)

Q3 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ f e d c b

Q4 = (A + B) $\overline{D}$ $\overline{E}$ $\overline{F}$ (H + h) $\overline{f}$ $\overline{e}$ $\overline{d}$ (b + a)

Q5 = (A + B) $\overline{D}$ $\overline{E}$ $\overline{F}$ g f e d c b

Q6 = B C D E F G H $\overline{f}$ $\overline{e}$ $\overline{d}$ (b + a)

Q7 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ (g + f) $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q8 = B C D E F G H $\overline{h}$ $\overline{g}$ $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q9 = $\overline{A}$ C D E F G H h g $\overline{f}$ $\overline{e}$ d c b

Q10 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ g $\overline{d}$ $\overline{c}$ (b + a)

Q11 = B $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ $\overline{h}$ $\overline{g}$ $\overline{f}$ $\overline{e}$ $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q12 = B C D E F $\overline{h}$ $\overline{g}$ $\overline{f}$ $\overline{e}$ $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q13 = (A + B) $\overline{D}$ (F + G) $\overline{h}$ $\overline{g}$ $\overline{f}$ $\overline{e}$ $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q14 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ $\overline{E}$ $\overline{F}$ h g f e d c b

Q15 = B C D E F G H $\overline{f}$ $\overline{e}$ $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q16 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ $\overline{E}$ $\overline{F}$ (H + h) $\overline{f}$ $\overline{e}$ $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q17 = B C D $\overline{H}$ $\overline{h}$ $\overline{g}$ e $\overline{c}$ $\overline{b}$ $\overline{a}$

Q18 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ $\overline{h}$ $\overline{g}$ c b

Q19 = C D E F G H h g f e d c

Q20 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ G H h g $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q21 = C D $\overline{G}$ $\overline{H}$ $\overline{h}$ $\overline{g}$ d c

Q22 = (A + B) $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ (g + f) $\overline{d}$ (b + a)

Q23 = $\overline{A}$ $\overline{B}$ $\overline{C}$ F G H h g f e d c b

Q24 = (A + B) $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ f e d c

Q25 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ $\overline{h}$ $\overline{g}$ $\overline{f}$ $\overline{e}$ $\overline{d}$ (b + a)

Q26 = $\overline{A}$ $\overline{B}$ E $\overline{H}$ $\overline{h}$ e $\overline{b}$ $\overline{a}$

Q27 = $\overline{A}$ $\overline{B}$ F (E + G) $\overline{h}$ $\overline{g}$ $\overline{f}$ $\overline{e}$ $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q28 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ $\overline{E}$ H h g $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q29 = $\overline{A}$ $\overline{B}$ $\overline{C}$ $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ $\overline{H}$ $\overline{h}$ e d c

Q30 = C D E $\overline{H}$ $\overline{h}$ $\overline{g}$ $\overline{f}$ $\overline{e}$ $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q31 = (A + B) $\overline{D}$ $\overline{E}$ $\overline{F}$ $\overline{G}$ (h + g) $\overline{e}$ $\overline{d}$ $\overline{c}$ $\overline{b}$ $\overline{a}$

Q32 = C $\overline{F}$ $\overline{G}$ $\overline{H}$ f e d

Q33 = C D E F G H h g $\overline{d}$ (b + a)

Table 3.2 (continued)

Q34 = $(A + B)\ \overline{D}\ G\ H\ h\ g\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q35 = $(A + B)\ \overline{D}\ \overline{E}\ \overline{F}\ h\ (H + g)\ \overline{d}\ (\overline{e} + \overline{c})\ (b + a)$

Q36 = $B\ C\ D\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ \overline{f}\ b\ (c + a)$

Q37 = $(A + B)\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ f\ (\overline{d} + \overline{c})\ (b + a)$

Q38 = $(A + B)\ (\overline{C} + \overline{D})\ F\ \overline{h}\ \overline{g}\ \overline{f}\ \overline{e}\ \overline{d}\ (b + a)$

Q39 = $B\ C\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q40 = $B\ C\ D\ E\ F\ G\ \overline{g}\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q41 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ G\ H\ h\ g\ f\ e\ d\ c\ b$

Q42 = $B\ C\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ \overline{f}\ b\ c$

Q43 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ H\ h\ g\ f\ e\ d\ c\ b$

Q44 = $(A + B)\ (\overline{C} + \overline{D})\ F\ G\ \overline{g}\ \overline{f}\ \overline{e}\ \overline{d}\ (b + a)$

Q45 = $(A + B)\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ d\ c\ b$

Q46 = $B\ C\ D\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ \overline{f}\ \overline{e}\ \overline{d}\ (b + a)$

Q47 = $(A + B)\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ g\ f\ (\overline{c} + \overline{d})\ (b + a)$

Q48 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ e\ d\ c\ \overline{a}$

Q49 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ e\ d\ c\ b\ a$

Q50 = $\overline{A}\ \overline{B}\ \overline{C}\ E\ F\ G\ \overline{h}\ \overline{g}\ e\ d\ c\ \overline{a}$

Q51 = $\overline{A}\ \overline{B}\ \overline{C}\ E\ F\ G\ \overline{h}\ \overline{g}\ e\ d\ c\ b\ a$

Q52 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ (G + H)\ (\overline{h} + \overline{g})\ (f + e)\ \overline{c}\ \overline{b}\ \overline{a}$

Q53 = $\overline{A}\ \overline{B}\ \overline{C}\ F\ G\ H\ h\ g\ f\ e\ d\ \overline{a}$

Q54 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ (h + g)\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q55 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ (H + h)\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q56 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ \overline{f}\ \overline{e}\ b$

Q57 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ (h + g)\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q58 = $\overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ F\ G\ H\ h\ g\ f\ e\ d\ \overline{b}\ \overline{a}$

Q59 = $(B + C)\ \overline{E}\ (\overline{D} + \overline{F})\ h\ (H + g)\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q60 = $C\ \overline{G}\ \overline{H}\ \overline{h}\ \overline{g}\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q61 = $\overline{A}\ \overline{B}\ \overline{C}\ F\ (E + G)\ \overline{h}\ \overline{g}\ \overline{f}\ c\ b$

Q62 = $B\ C\ \overline{F}\ \overline{G}\ \overline{H}\ (e + g)\ f\ \overline{c}\ \overline{b}\ \overline{a}$

Q63 = $B\ (\overline{D} + \overline{E})\ G\ \overline{g}\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

Q64 = $\overline{A}\ \overline{B}\ E\ \overline{G}\ \overline{H}\ f\ e\ (\overline{d} + \overline{c})\ (b + a)$

Q65 = $\overline{A}\ (C + D)\ \overline{F}\ (\overline{E} + \overline{G})\ h\ g\ f\ (\overline{d} + \overline{c})\ (b + a)$

Table 3.2 (continued)

$Q66 = (\overline{A} + \overline{B})\ E\ F\ G\ \overline{g}\ (\overline{h} + \overline{f})\ d\ c\ (e + b)$

$Q67 = C\ D\ E\ F\ G\ H\ \overline{g}\ \overline{f}\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

$Q68 = \overline{A}\ \overline{B}\ \overline{C}\ \overline{D}\ \overline{E}\ \overline{F}\ \overline{G}\ (h + g)\ \overline{e}\ \overline{d}\ \overline{c}\ \overline{b}\ \overline{a}$

$Q69 = B\ C\ D\ E\ (\overline{F} + \overline{G})\ H\ (\overline{h} + \overline{g})\ e\ d\ c\ b\ a$

$Q70 = \overline{A}\ \overline{B}\ E\ F\ G\ H\ (\overline{h} + \overline{g})\ (f + e)\ \overline{c}\ \overline{b}\ \overline{a}$

Table 3.3 Recognition logics.

1	1_1	I	Q1	
		II	Q2	
	1_2	I	Q3	
		II	Q1	
2	2_1	I	Q4	inhibition by B_1, 8_1
		II	Q5	
	2_2	I	Q6	
		II	Q4	
3	3_1	I	Q4	inhibition by B_1.
		II	Q2	
	3_2	I	Q1	
		II	Q4	
4	4_1	I	Q7	inhibition by H_1
		II	Q8	
	4_2	I	Q9	
		II	Q7	
5	5_1	I	Q4	inhibition by 8_1, B_1
		II	Q6	
	5_2	I	Q5	
		II	Q4	
6	6_1	I	Q10	inhibition by 1_1, W_1
		II	Q1	
7	7_1	I	Q5	
		II	Q11	
	7_2	I	Q12	
		II	Q5	
8	8_1	I	Q6	
		II	Q6	
9	9_1	I	Q1	inhibition by T_1
		II	Q13	
0	0_1	I	Q2	
		II	Q1	
	0_2	I	Q1	
		II	Q2	

Table 3.3 (continued)

		I	Q1	
		II	Q3	
		I	Q12	
		II	Q1	
		I	Q14	inhibition by N_1, W_1
		II	Q15	
		I	Q15	
		II	Q14	
		I	Q16	inhibition by H_1
		II	Q14	
		I	Q1	
		II	Q16	
		II	Q1	
A	A_1	I	Q17	
		II	Q18	
	A_2	I	Q18	
		II	Q17	
B	B_1	I	Q4	inhibition by 8_1
		II	Q1	
	B_2	I	Q19	
		II	Q4	
C	C_1	I	Q2	inhibition by O_1 2
		II	Q20	
	C_2	I	Q2	
		II	Q2	
D	D_1	I	Q21	
		II	Q1	
	D_2	I	Q20	
		II	Q2	
E	E_1	I	Q4	
		II	Q1	
	E_2	I	Q2	
		II	Q4	

Table 3.3 (continued)

F	F_1	I	Q13	inhibition by $_1$, R_1
		II	Q1	
G	G_1	I	Q22	inhibition by 8_1, B_1
		II	Q23	
	G_2	I	Q24	
		II	Q22	
H	H_1	I	Q16	inhibition by N_2
		II	Q1	
	H_2	I	Q1	
		II	Q16	
I	I_1	I	Q1	inhibition by D_1
		II	Q2	
	I_2	I	Q2	
		II	Q1	
J	J_1	I	Q25	inhibition by U_1, W_1
		II	Q3	
	J_2	I	Q1	
		II	Q25	
K	K_1	I	Q26	
		II	Q1	
	K_2	I	Q2	
		II	Q26	
L	L_1	I	Q25	inhibition by 1_1, U_2, W_1, 1
		II	Q1	
M	M_1	I	Q27	inhibition by N_1
		II	Q1	
	M_2	I	Q1	
		II	Q27	
N	N_1	I	Q28	
		II	Q1	
	N_2	I	Q29	
		II	Q30	

Table 3.3 (continued)

O	O_1	I	Q2	inhibition by O_1
		II	Q20	
	O_2	I	Q20	
		II	Q2	
P	P_1	I	Q31	inhibition by $_1$
		II	Q1	
Q	Q_1	I	Q32	
		II	Q23	
	Q_2	I	Q33	
		II	Q32	
R	R_1	I	Q34	inhibition by P_1
		II	Q1	
S	S_1	I	Q35	
		II	Q36	
	S_2	I	Q37	
		II	Q38	
T	T_1	I	Q1	
		II	Q39	
	T_2	I	Q39	
		II	Q1	
U	U_1	I	Q25	inhibition by W_1
		II	Q1	
	U_2	I	Q1	
		II	Q25	
V	V_1	I	Q18	inhibition by IIQ_1
		II	Q40	
W	W_1	I	Q41	
		II	Q1	
	W_2	I	Q1	
		II	Q41	
X	X_1	I	Q20	inhibition by IIQ_1, $\%_1$
		II	Q42	
	X_2	I	Q42	
		II	Q20	

Table 3.3 (continued)

Y	Y_1	I	Q43	inhibition by $_1$, $_1$, IIQ1
		II	Q39	
Z	Z_1	I	Q44	inhibition by S_2
		II	Q45	
	Z_2	I	Q46	
		II	Q47	
▪		I	-	inhibition by Q49, Q50, Q51,
		II	Q48	Q53, Q58, Q1 of II.
▮		I	-	inhibition by Q1, Q51 of II
		II	Q49	
▪ ▪		I	-	inhibition by Q1, Q51 of II
		II	Q50	
▪ ▮		I	-	inhibition by Q1
		II	Q51	
-	$-_1$	I	Q52	inhibition by Q50, Q51, Q58
		II	Q52	of II
+	$+_1$	I	Q53	
		II	Q54	
	$+_2$	I	Q54	
		II	Q53	
/	$/_1$	I	Q55	inhibition by IIQ1, V_1, X_1
		II	Q56	
⋈	$⋈_1$	I	Q57	
		II	Q58	
	$⋈_2$	I	Q58	
		II	Q57	
▽	$▽_1$	I	Q59	
		II	Q60	
{	$\{_1$	I	Q1	
		II	Q16	
	$\{_2$	I	Q2	
		II	Q1	

Table 3.3 (continued)

}	$}_1$	I	Q1	inhibition by 1_2
		II	Q2	
	$}_2$	I	Q16	
		II	Q1	
%	$%_1$	I	Q61	
		II	Q62	
?	$?_1$	I	Q63	inhibition by IIQ1
		II	Q64	
&	$&_1$	I	Q65	
		II	Q66	
	$&_2$	I	Q66	
		II	Q65	
▮		I	-	inhibition by Q1, Q59 of II
		II	Q67	
—	$—_1$	I	Q68	inhibition by Q53 of II
		II	Q68	
⇆	$⇆_1$	I	Q69	
		II	Q70	

Q16, Q17, Q18, Q19, Q20, Q21, Q22, Q24, Q25, Q26, Q27, Q28, Q29, Q31, Q32, Q33, Q34, Q35, Q37, Q39, Q41, Q42, Q43, Q44, Q46, Q52, Q53, Q54, Q55, Q57, Q58, Q59, Q61, Q63, Q65, Q66, Q68, Q69

The total number of these is 51,

(b) The characteristic feature logics for the input part II : Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q11, Q13, Q14, Q15, Q16, Q17, Q18, Q20, Q22, Q23, Q25, Q26, Q27, Q30, Q32, Q36, Q38, Q39, Q40, Q41, Q42, Q45, Q47, Q48, Q49, Q50, Q51, Q52, Q53, Q54, Q56, Q57, Q58, Q60, Q62, Q64, Q65, Q66, Q67, Q68, Q70,

The total number of these is 49.

In consequence the total number of logic functions is 100.

3.2.3 Considerations

The features of this principle of the recognition with two input parts are ;

(1) the machine works asynchronously, but it has a function of synchronous character where it is needed,

(2) the machine has few effects from noise by the inherent nature of the recognition principle,

(3) the characteristic feature logics are almost peculiar to respective characters, so that the need for the inhibition logics is not strong,

(4) the number of the characteristic feature logics for the input channels I and II is not so many as is imagined,

(5) a bad point is a trouble of knowing in design the ef-

fects from the adjacent words, because there might be a time when two characters are on the input parts simultaneously.

This method is neither realized in actual device, nor is its experiment done, but the system seems very good for the characters which are rather difficult to be treated by the principle of Chapter 2. The realization of two input parts with many channels may be hard by photo transistors, but may be very easily achieved by silicon P-N junction (solar battery).

3.3 Recognition Method with Two Dimensional Input Part

3.3.1 Principle

From the above discussion the input part with two vertical lines seems to be more powerful than that with one vertical line. To extend this idea to the n line input part, we reach to the two dimensional input part. To have a picture of a pattern with a spot, it must be moved horizontally and vertically on the two dimensional space. With a line the scanning in one direction covers the surface. With a two dimentional input part, the scanning is not necessary and a picture of a pattern is received in a moment. Therefore this eliminates the memory circuits for certain operations, which are needed in the scanning by a spot or by a line element. The two dimensional input part is feasible by silicon P-N junction (solar battery) where a cell can be made 1 mm × 1 mm.

The design is done for 10 numerals and four special sym-

bols of OCR-A.[13] These characters can be roughly represented by 9 × 5 meshes as shown in Fig. 3.4. For the distinction of these, 8 peepholes are set up as shown in Fig. 3.5. The locations are indicated by 1, 2, ..., 8, 9 from top to bottom, and A, B, C, D, E from right to left. Then the eight peepholes are expressed by the following logical functions.

$$P1 = A3 \cdot A2 \cdot A1 \cdot B1 \cdot C1 \cdot D1 \cdot E1$$

$$P2 = C2 \cdot C3 \cdot C4 \cdot C5 \cdot C6 \cdot C7 \cdot C8$$

$$P3 = A7 \cdot A8 \cdot A9 \cdot B9 \cdot C9 \cdot D9$$

$$P4 = B6 \cdot C6 \cdot D6 \cdot E6 \cdot E5 \cdot E4 \cdot E3 \cdot E2$$

$$P5 = A9 \cdot B9 \cdot C9 \cdot D9 \cdot E9 \cdot E8 \cdot E7$$

$$P6 = A2 \cdot A3 \cdot A4 \cdot A5 \cdot A6 \cdot A7 \cdot A8$$

$$P7 = B1 \cdot C1 \cdot D1 \cdot B9 \cdot C9 \cdot D9$$

$$P8 = A2 \cdot A3 \cdot A4 \cdot A5 \cdot B5 \cdot C5 \cdot D5 \cdot E5$$

When these logical conditions are applied to the characters, Table 3.4 is obtained, in which "o" indicates the satisfaction of the logical condition, "x" indicates the contrary, and "-" indicates the arbitrary state. From this table the following recognition logic functions Ri can be written.

$$R0 = P1\ \overline{P2}\ P3\ \overline{P4}\ P5\ P6\ P7\ \overline{P8}$$

$$R1 = \overline{P1}\ P2\ P3\ \overline{P4}\ \overline{P5}\ \overline{P6}\ \overline{P8}$$

$$R2 = P1\ \overline{P2}\ \overline{P3}\ \overline{P4}\ P5\ \overline{P6}\ P7\ P8$$

$$R3 = P1\ \overline{P2}\ P3\ \overline{P4}\ \overline{P5}\ P6\ P7\ \overline{P8}$$

$$R4 = \overline{P1}\ \overline{P2}\ \overline{P3}\ P4\ \overline{P5}\ \overline{P6}\ \overline{P7}\ \overline{P8}$$

$$R5 = \overline{P1}\ \overline{P2}\ P3\ \overline{P4}\ \overline{P5}\ \overline{P6}\ P7\ \overline{P8}$$

$$R6 = \overline{P1}\ \overline{P2}\ P3\ P4\ P5\ \overline{P6}\ \overline{P7}\ \overline{P8}$$

$$R7 = P1\ \overline{P2}\ \overline{P3}\ \overline{P4}\ \overline{P5}\ \overline{P6}\ \overline{P7}\ \overline{P8}$$

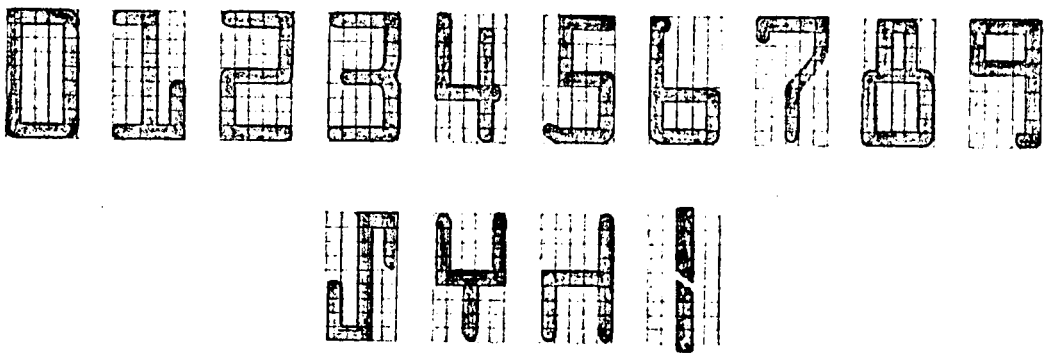


Fig. 3.4 OCR-A characters which are drawn on 9 x 5 mesh.

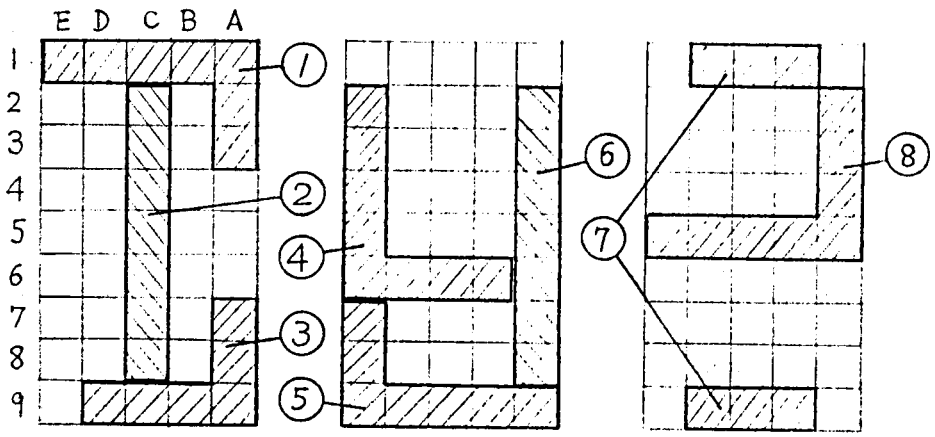


Fig. 3.5 Eight peepholes for the recognition of OCR-A numerals and three special symbols.

	P1	P2	P3	P4	P5	P6	P7	P8
0	o	x	o	x	o	o	o	x
1	x	o	o	x	x	x	-	x
2	o	x	x	x	o	x	o	o
3	o	x	o	x	x	o	o	x
4	x	x	x	o	x	x	x	x
5	x	x	o	x	x	x	o	x
6	x	x	o	o	o	x	x	x
7	o	x	x	x	x	x	x	x
8	x	x	o	x	o	x	o	x
9	o	x	x	x	x	o	x	x
ſ	x	o	x	x	x	x	x	x
Y	x	x	x	x	x	x	x	o
ſ	x	x	x	x	x	o	x	o

Table 3.4 Relation of logical conditions and OCR-A font.

$$R8 = \overline{P1}\ \overline{P2}\ P3\ \overline{P4}\ P5\ \overline{P6}\ P7\ \overline{P8}$$

$$R9 = P1\ \overline{P2}\ \overline{P3}\ \overline{P4}\ \overline{P5}\ P6\ \overline{P7}\ \overline{P8}$$

$$R_{J} = \overline{P1}\ P2\ \overline{P3}\ \overline{P4}\ \overline{P5}\ \overline{P6}\ \overline{P7}\ \overline{P8}$$

$$R_{\downarrow} = \overline{P1}\ P2\ \overline{P3}\ \overline{P4}\ \overline{P5}\ \overline{P6}\ \overline{P7}\ P8$$

$$R_{\downarrow} = \overline{P1}\ \overline{P2}\ \overline{P3}\ \overline{P4}\ \overline{P5}\ P6\ \overline{P7}\ P8$$

The input patterns may move horizontally or vertically relative to the two dimensional input part. Here the horizontal movement is supposed. Then an input pattern moving under the input part produces the recognition output when it comes just under the prescribed position. An important point is there must be a guarantee that no output appears in other positions during the movement of a pattern. This is examined one by one for the ideal patterns, and also was guaranteed by the computer simulation of this system for several sample patterns. The simulation was done by the programming system explained in Chapter 4. Thus this system is also an asynchronous one, and has several excellent points that the asynchronous system intrinsically has.

The eight peepholes are determined by the following reasons.

(i) If a peephole consists of only one mesh, the effect of noise is great.

(ii) The proper time when an input pattern comes just under the prescribed position can be known more exactly if there are some peepholes which are angular.

(iii) All the cells composing the peepholes are taken for the existence of patterns, and there is no cell which demands

the non-existence of a portion of patterns. Thus the extraneous ink has no effect on the recognition functions. The void may decrease the recognition rate, but the threshold value of input circuits can be set to the value where voids seldom appear.

(iv) The number of peepholes is to be as few as possible.

3.3.2 The Recognition System

The block diagram of the system is shown in Fig. 3.6. The two dimensional input part is composed of three parts as shown in Fig. 3.7. The part [a] is provided for the determination of the vertical position of a character entering to the input part. The part [b] is connected to the decision logics. And the part [c] is provided for the recognition timing and for the reset of all the circuits to the initial state. The input pattern is supposed to be the size of 9 × 5 meshes, while the part [b] has 17 × 5 meshes. These 17 meshes in the vertical direction are provided for the reason that 9 meshes are for an input character, 6 meshes are for the vertical shift of an input pattern when more than 30 lines are fed, and 2 meshes are for the inclination of a paper setting.

The hold circuit connected to each cell of the input part [a] detects and holds the input signal of a pattern until the pattern has passed completely through the input part [a]. At this time the information in the hold circuits is transferred to the channel detection logic, which determines the position of an input character. This position information is used by

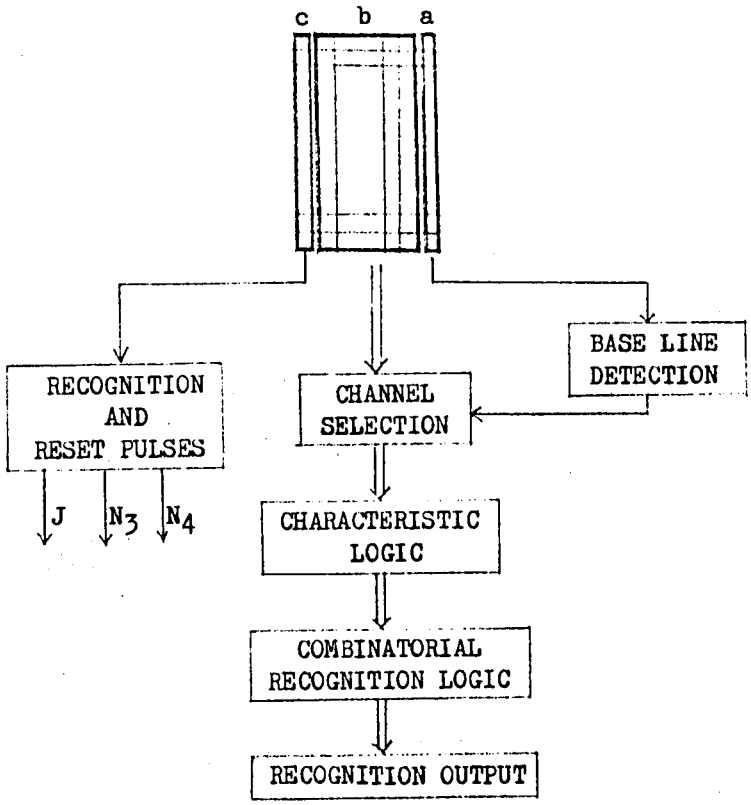


Fig. 3.6 Block diagram of a recognition machine with two dimensional input part.

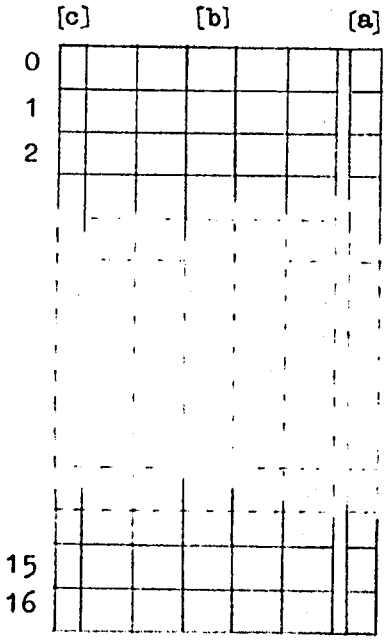


Fig. 3.7 Two dimensional input part with two auxiliary channels.

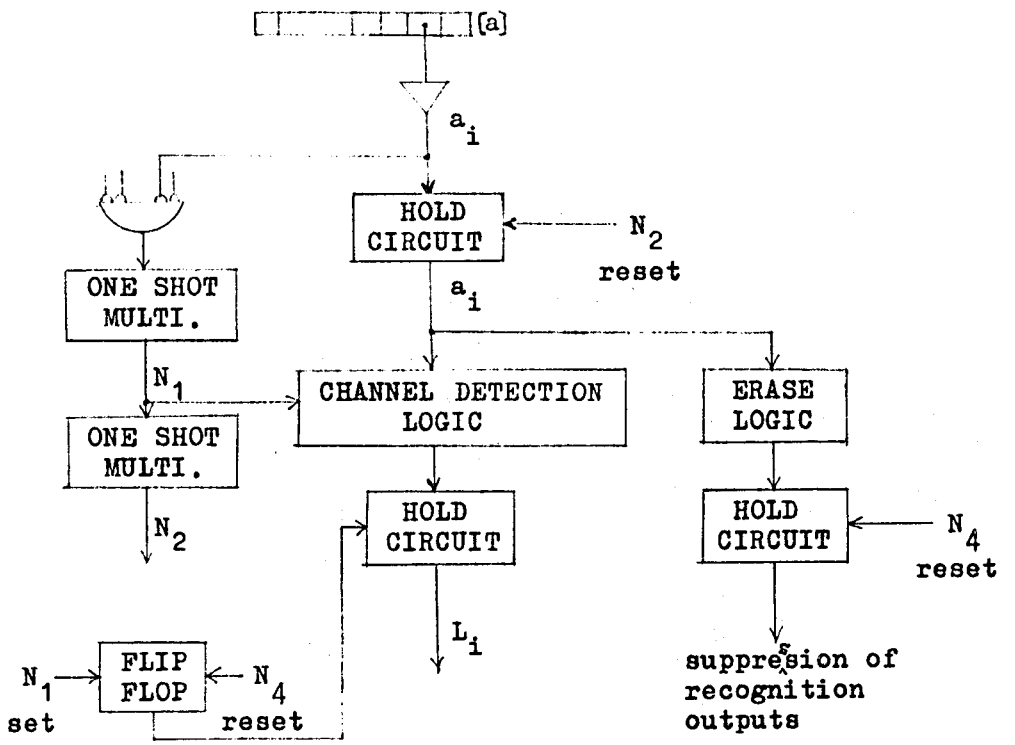


Fig. 3.8 Input part [a] and its related circuits.

the channel exchange which selects the proper 9 × 5 meshes from the 17 × 5 meshes.

The next recognition circuits are the combinatorial logic functions mentioned above.

The input part [a] to the hold circuits of the channel detection logics is shown in Fig. 3.8. The cells of input part [a] are named a_0, a_1, $\cdots a_{16}$ from top to bottom. The hold circuit is shown in Fig. 3.9. The pulse N_2 is obtained from the circuit of Fig. 3.8, where N_1 is a short pulse when the condition $\overline{a}_0\overline{a}_1\cdots\overline{a}_{16} = 1$ is satisfied, i.e. an input pattern has completely passed away from the input part [a], and where N_2, a short pulse, follows just after N_1. This pulse N_2 cancels the information stored in the hold circuit of Fig. 3.9, after it is sent to the channel detection logics shown in Fig. 3.10 by the pulse N_1, and the circuit is prepared for the next input pattern. Here the logic functions $L_1', \cdots L_8'$ are the followings.

$$L_1' = a_1 \cdot a_2 \cdots a_8 \cdot \overline{a}_{10} \cdot \overline{a}_{11} \cdot \overline{a}_{12}$$

$$L_2' = \overline{a}_0 \cdot a_2 \cdot a_3 \cdot \cdots a_9 \cdot \overline{a}_{11} \cdot \overline{a}_{12} \cdot \overline{a}_{13}$$

$$L_3' = \overline{a}_0 \cdot \overline{a}_1 \cdot a_3 \cdot a_4 \cdot \cdots a_{10} \cdot \overline{a}_{12} \cdot \overline{a}_{13} \cdot \overline{a}_{14}$$

$$L_4' = \overline{a}_0 \cdot \overline{a}_1 \cdot \overline{a}_2 \cdot a_4 \cdot a_5 \cdot \cdots a_{11} \cdot \overline{a}_{13} \cdot \overline{a}_{14} \cdot \overline{a}_{15}$$

$$L_5' = \overline{a}_1 \cdot \overline{a}_2 \cdot \overline{a}_3 \cdot a_5 \cdot a_6 \cdot \cdots a_{12} \cdot \overline{a}_{14} \cdot \overline{a}_{15} \cdot \overline{a}_{16}$$

$$L_6' = \overline{a}_2 \cdot \overline{a}_3 \cdot \overline{a}_4 \cdot a_6 \cdot a_7 \cdot \cdots a_{13} \cdot \overline{a}_{15} \cdot \overline{a}_{16}$$

$$L_7' = \overline{a}_3 \cdot \overline{a}_4 \cdot \overline{a}_5 \cdot a_7 \cdot a_8 \cdot \cdots a_{14} \cdot \overline{a}_{16}$$

$$L_8' = \overline{a}_4 \cdot \overline{a}_5 \cdot \overline{a}_6 \cdot a_8 \cdot a_9 \cdot \cdots a_{15}$$

These logic functions are decided considering the case that the character height might be 8, 9 or 10 meshes.

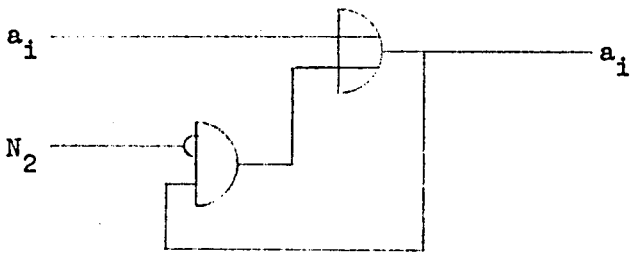


Fig. 3.9 Hold circuit of Fig. 3.8.

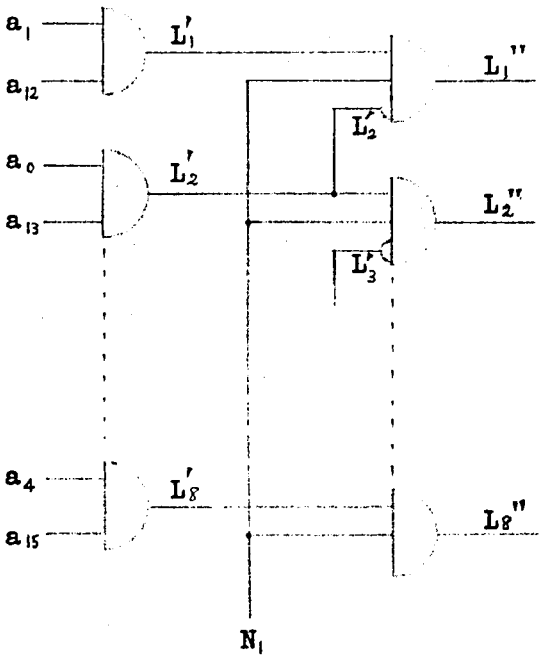


Fig. 3.10a Channel detection logics.

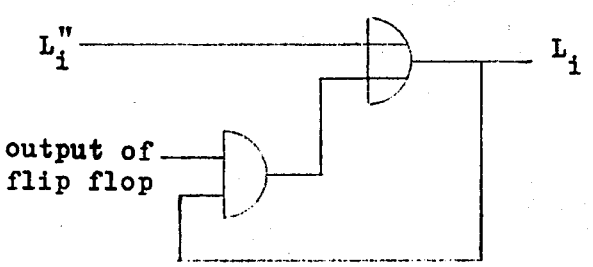


Fig. 3.10b L_i-channel hold circuit.

Therefore the base channel is determined by the following logic functions, which are gated by the short pulse N_1 and the information is transferred to the hold circuit of Fig. 3.10b.

$$L_1 = L_1' \cdot \overline{L}_2' \cdot N_1$$
$$L_2 = L_2' \cdot \overline{L}_3' \cdot N_1$$
$$L_3 = L_3' \cdot \overline{L}_4' \cdot N_1$$
$$L_4 = L_4' \cdot \overline{L}_5' \cdot N_1$$
$$L_5 = L_5' \cdot \overline{L}_6' \cdot N_1$$
$$L_6 = L_6' \cdot \overline{L}_7' \cdot N_1$$
$$L_7 = L_7' \cdot \overline{L}_8' \cdot N_1$$
$$L_8 = L_8' \cdot N_1$$

This hold circuit is maintained by the output of flip-flop in Fig. 3.8, which is activated by the pulse N_1 and is reset by the pulse N_4.

All the outputs from the cells of the input part [b] are connected to the 9×5 two dimensional lines by the information from the channel detection circuit. In Fig. 3.11 the combinatorial logic circuit for the A column is shown. For the other columns the circuit is the same.

Fig. 3.12 is for the input part [c], whose output generates a short pulse N_3 and after this a short pulse N_4. From the output of input part [c], the hold circuit J is activated, which has output as long as $\overline{c}_0 \cdot \overline{c}_1 \cdot \overline{c}_2 \cdots\cdots \overline{c}_{16} = 1$ is satisfied. This output gates the combinatorial recognition logics so as not to work during the transient period between characters. The timing relations among these are shown in Fig. 3.13. The long vertical line is used as the erase mark of a

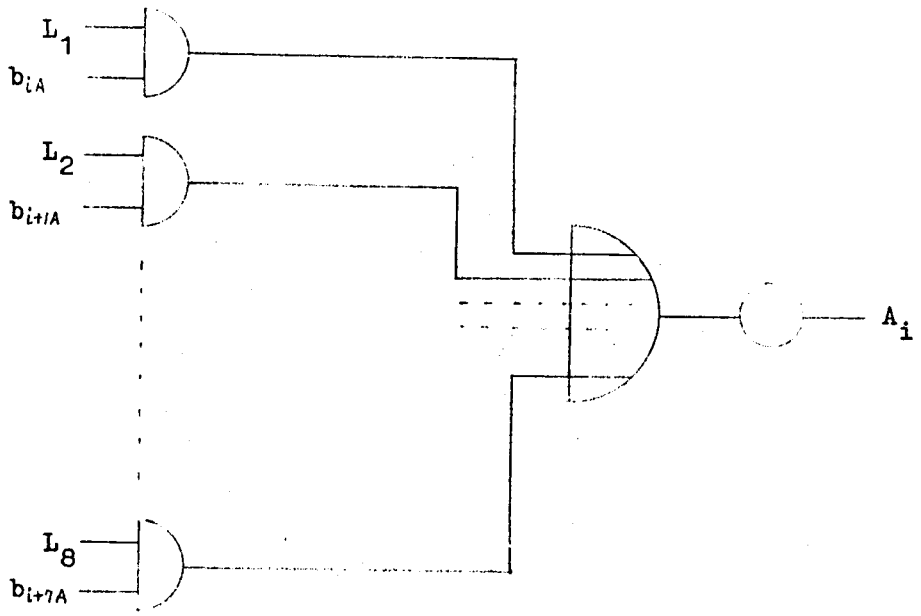


Fig. 3.11 Combinatorial logic circuit of channel selection.

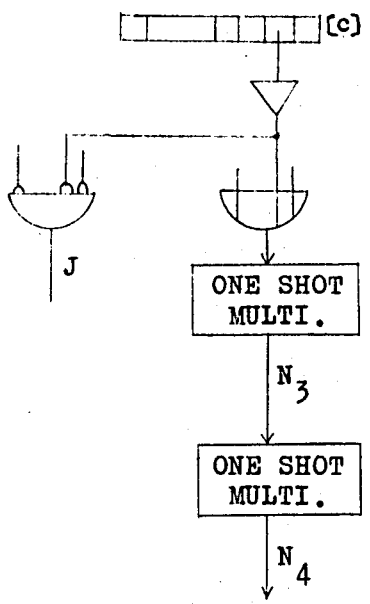


Fig. 3.12 Input part (c) and its related circuits.

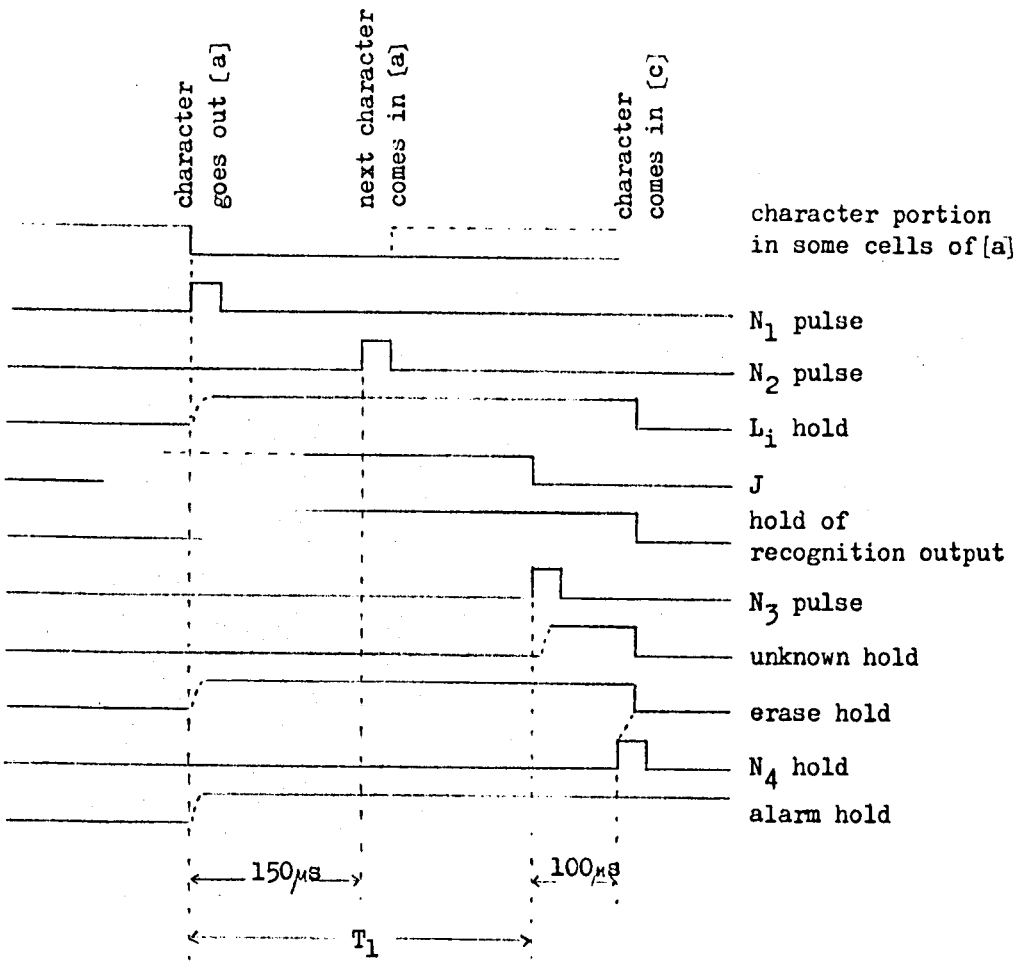


Fig. 3.13 Timing relations among many outputs.

This is written by the character speed of 1000 mm/s.

T_1 is 100 --500μs.

Fig. 3.14 Long vertical line used as erase mark.

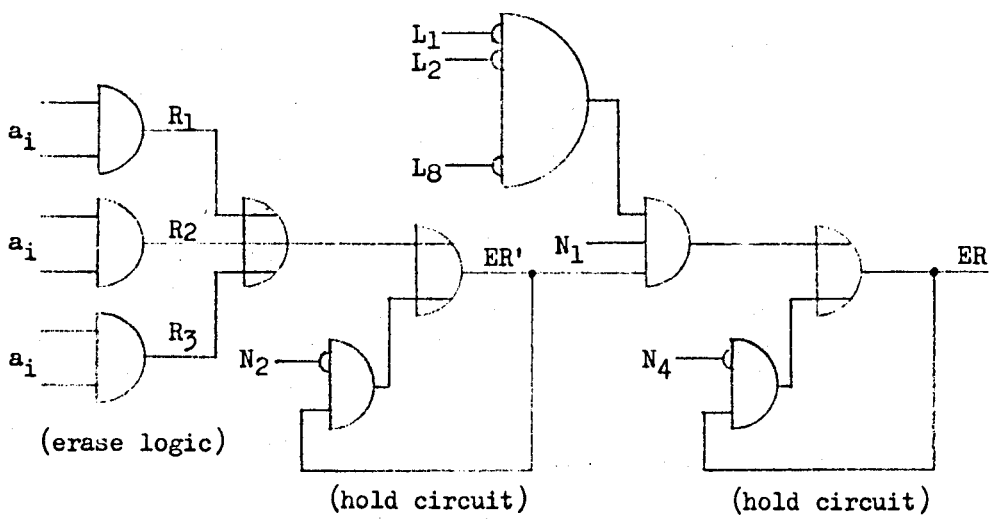


Fig. 3.15 Erase circuit.

character and is typed on the character as shown in Fig. 3.14. The detection of erase sign is done by the following logic.

$$ER' = R_1 + R_2 + R_3$$

where

$$R_1 = (a_0 + a_1)(a_2 + a_3)\cdots\cdots(a_{12} + a_{13})$$

$$R_2 = (a_2 + a_3)(a_4 + a_5)\cdots\cdots(a_{14} + a_{15})$$

$$R_3 = (a_4 + a_5)(a_6 + a_7)\cdots\cdots(a_{14} + a_{15})a_{16}$$

The circuit of the erase sign ER is obtained from the output ER' as shown in Fig. 3.15. Therefore the reject signal for the unknown input pattern is obtained by the condition that there is no recognition output, no erase output, and there is an output from the channel detection logics, all of which are gated by the signal N_3.

This recognition system is not constructed but was simulated by a digital computer for many input samples which are properly digitized. The result of the simulation showed the stable performance. This system has no complicated timing control so that the operation is very stable.

CHAPTER 4

PROGRAMMING SYSTEM FOR THE SIMULATION OF LOGICAL CIRCUITS

4.1 Introduction

When a large system is to be constructed, the design of the system is to be reexamined from all possible standpoints. Among them most powerful method is the simulation of the system as it works actually. The recent development of an electronic digital computer has enabled the simulation of a very large system.[16] Accordingly programming systems such as SIMSCRIPT,[17] which aims at easy programming for the simulation, have been being developed.

The author has developed a very simple but useful programming system for the simulation of logical circuits. This system has been developed mainly for the simulations of character recognition machines presented in Chapters 2 and 3. The system simulation was very successful for the determination of many characteristic feature logics and for the error findings. This system can simulate the performance of asynchronous logical circuits as well as synchronous one, and many other binary operated pulse circuits. In the system are included such functions as AND gate, OR gate, delay line, trigger circuit, flip-flop, switch, and control instructions such as go to, test equal etc.. The system is composed of two parts. One is a source program written in pseudo instructions defined by the system, and the other is an assembler which converts the source program into machine instruction program,

and punches out on a paper tape. Seventeen kinds of pseudo instructions are provided for, each of which is corresponded a series of machine instructions.

The block diagram of the programming system is shown in Fig. 4.1. The assembler is a 550 words program. The system

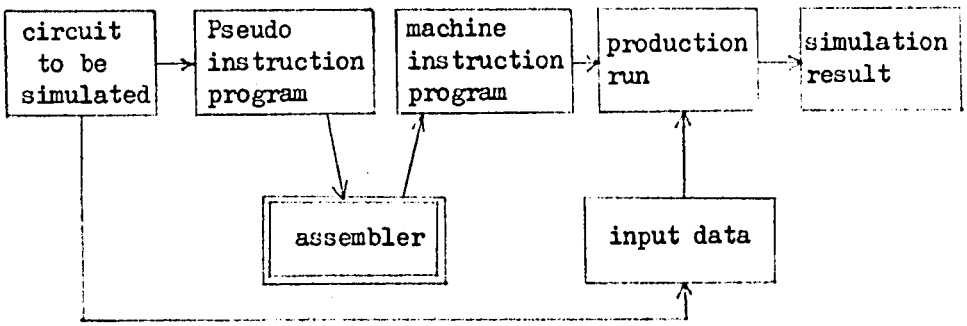


Fig. 4.1 Block diagram of the programming system.

has also a compiler of Boolean expressions composed of AND, OR, and NOT. Since these Boolean expressions constitute an operator language investigated in Chapter 2, PART I, the structure of the compiler is explained in relation to the results obtained in that chapter.

This system is implemented in the digital computer KDC-I (Kyoto University Digital Computer), whose main memory is a magnetic drum of 4200 words. Since the memory size is small, the assembler punches out the assembled machine instructions each time a pseudo instruction is read in.

4.2 Pseudo Instructions

Pseudo instructions are generally of the form that first comes the name of the instruction, next a delimiter "δ", then

numbers and /or labels which are delimited by a special symbol "||". These symbols δ and || are tape control codes and have special functions by KDC-I. Numbers are limited to positive and less than 9999. Labels are composed of less than 5 characters and the top symbol of labels must be an alphabetic character. To the head of labels the symbol "−" can be attached, which means logical NOT of the variables represented by the labels.

4.2.1 Pseudo Instructions of Logical Circuits

1. AND δ n || A || B || · · · || Z ||

The logical AND of n input lines A, B, · · · is performed and the output is stored in Z. The corresponding machine instructions are as follows.

```
ADD/     A  ┐
AND      B  │
· · · · ·   │  } n
· · · · ·   │
            │
AND         ┘
STO      Z
```

These are written by the instructions of KDC-I, and are explained in Appendix, but for the complete reference see the bibliographies(18) and (19).

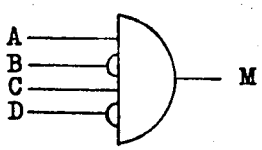


Fig. 4.2 AND gate. M = A·(-B)·C·(-D)

For example the AND gate of Fig. 4.2 is represented by

AND § 4 ‖ A ‖ -B ‖ C ‖ -D ‖ M ‖

The machine instructions generated for this pseudo instruction are,

```
ADD/    B
NOT
STO     W1
ADD/    D
NOT
STO     W2
ADD/    A
AND     W1
AND     C
AND     W2
STO     M
```

W1 and W2 are the working storages provided by the assembler.

2. OR § n ‖ A ‖ B ‖ · · · · ‖ Z ‖

The logical OR of n input lines A, B, · · · is performed and the output is stored in Z. The corresponding machine instructions are as follows.

```
ADD/    A  ⎫
IOR     B  ⎪
· · · · · ·  ⎬ n
IOR        ⎪
STO     Z  ⎭
```

3. DEL δ n ‖ A ‖ B ‖

The signal on the input line A is delayed n bits time and the output is obtained on the line B.

The corresponding machine instructions are as follows.

ADD/	$A^{(n)}$
STO	B
ADD/	$A^{(n-1)}$
STO	$A^{(n)}$
...	.. ..
...	.. ..
...	.. ..
ADD/	A
STO	$A^{(1)}$

$A^{(1)}, \cdots, A^{(n)}$ are the working storages provided by the assembler.

4. TRIG δ A ‖ B ‖

Trigger pulse is obtained on the line B according as the signal on the line A changes from 0 to 1 or 1 to 0, in which case the signal on B is + 1 or - 1 respectively. The output on the line B is to be connected directly to the flip-flop. The output + 1 or - 1 is cleared away at the next bit time. The corresponding machine instructions are as follows.

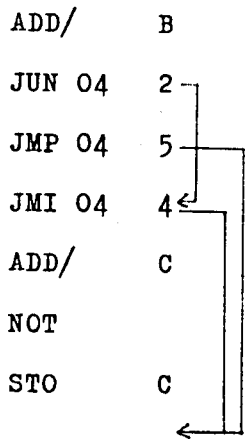
ADD/	A
SUB	A'
STO	B
ADD/	A

```
STO     A'
```

A' contains the content of 1 bit time before of A.

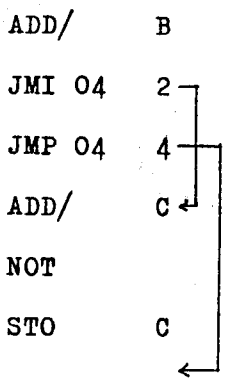
5. FF + δ B ‖ C ‖

Trigger B turns the state of the flip-flop C, only when the polarity of the trigger is positive. The corresponding machine instructions are as follows.



6. FF - δ B ‖ C ‖

The same as FF +, except that the trigger B is negative. The corresponding machine instructions are as follows.



4.2.2 Pseudo Instructions of Input and Output

7. READ δ n ‖ A ‖ B ‖ · · · ·

From the photo tape reader n data are read in and stored

in n storages A, B, $\cdots$ in this order. The value of each datum is either 1 or 0 and the data tape corresponding to this instruction is of the form,

$$\underbrace{101001\cdots}_{n}\|$$

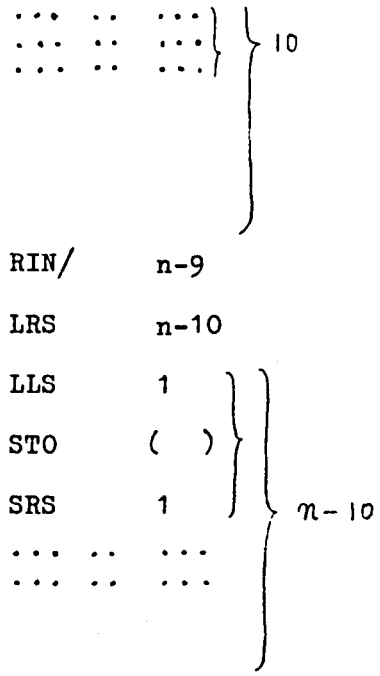
This corresponds to all the states of input lines A, B, $\cdots$ at a certain bit time. The corresponding machine instructions are as follows.

For $n \leq 10$;

```
RIN/     n + 1
LRS      n
LLS      1   }
STO      A   }
SRS      1   }
LLS      1   }
STO      B   }   } n
SRS      1   }
... ..   ..  }
... ..   ..  }
```

For $11 \leq n \leq 20$;

```
RIN/     10
LRS      10
LLS      1   }
STO      A   }
SRS      1   }
```



8. DEF δ A ‖ x ‖ B ‖ y ‖ ····

This defines A as x, B as y, ····. x, y etc. are to be 1 or 0. This corresponds to manual switches as shown

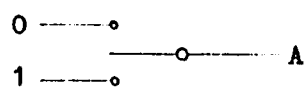


Fig. 4.3 Manual switch corresponding to the function DEF.

in Fig. 4.3. Such switch names and values can be written as many as one wishes to the right of DEF δ. The corresponding machine instructions are as follows.

RAA/ x
STO A
RAA/ y

```
STO      B
...  ..  ...
...  ..  ...
```

9. WRIT & A || B || · · ·

The contents of A, B, · · · are printed out by the following format.

```
A     *
B     *
..    ..
..    ..
```

where * is either 1 or 0. The labels A, B, · · · can be written as many as one wishes. The corresponding machine instructions are as follows.

```
WSP      CR1   )
ADD/     (A)   |
WRT      1005  |
WSP      SP1   |
ADD/     A     |
SLS      10    |
WRT      1     )
WSP      CR1   )
ADD/     (B)   |
...  ..  ..    |
...  ..  ..    )
```

(A) or (B) means the labels themselves are stored in the locations (A) or (B).

4.2.3 Pseudo Instructions of Controls

10. BEGIN δ n ‖

This pseudo instruction is to be written at the top of a simulation program, in which n is the top location of the assembled machine instruction program, and the program starts from that location. n must be larger than 400. When the assembler reads this instruction, it prepares to read the input data and to print out the results. The corresponding machine instructions are as follows.

```
      δ
SEX   10   n
  4δ
SEL        PR
SEL        PTR
RAA/       1
PAM
```

11. CLE δ m ‖ n ‖

The memory storages from m to n (m<n) are cleared to 0. This is required very often when the initial conditions of many labels are zero. Usually this instruction is written next to the pseudo instruction BEGIN δ.

The corresponding machine instructions are as follows.

```
SEX   10   n
SEX   20   m
STO/  20   0
JXU   24   9999
```

12. TEQ $ A || B || L ||

The contents of A and B are compared each other, and if they are not equal the control jumps to the label L. This is conveniently used to test whether an asynchronous circuit reached the stable state or not. The corresponding machine instructions are as follows.

```
ADD/    A
SUB     B
PSX     (L)
JNZ     0
```

Here (L) means the location in which the address L is stored.

13. GO $ L ||

Control jumps to the label L unconditionally.. The corresponding machine instructions are

```
PSX     (L)
JMP     0
```

(L) has the same meaning as explained in TEQ.

14. IF $ A || L ||

If the content of A is not zero, control jumps to L. The negation symbol "-" can be attached to the label A. The corresponding machine instructions are as follows.

```
ADD/    A
PSX     (L)
JUN     0
```

15. L $ T ||

T is a label to which the instructions such as GO, IF,

TEQ transfer the control. This can be attached to the head of any instruction except FF + and FF -. There is no corresponding machine instruction. Only the label T is stored in the jump table.

16. HJM δ

The computer halts at this pseudo instruction and by pushing the start button it resumes the operation. The corresponding instruction is as follows.

```
HJM   04   1
```

17. FINAL δ

This must be attached to the end of a simulation program, which tells the assembler the end of a program. The assembler punchs out the jump table and the label tabel which are necessary for the machine program. These are punched out in the following format.

```
HJM   04   0
      δ
SEX   10   100
      4δ
   ┌ ─ ─ ─ ─ ┐
   │  jump   │
   │  table  │
   └ ─ ─ ─ ─ ┘
      δ
SEX   10   200
      4δ
```

4.3 Concept of Timing

By the simulation of a system in computer, the parallel processing in the system must be followed in the serial manner, so that how to change the parallel processing to the serial one is an important point. If it is done in an arbitrary manner, wrong results will come out.

4.3.1 Asynchronous Circuits

Let us consider, for example, the performance of a logical circuit shown in Fig. 4.4a, where a group of pulses shown in

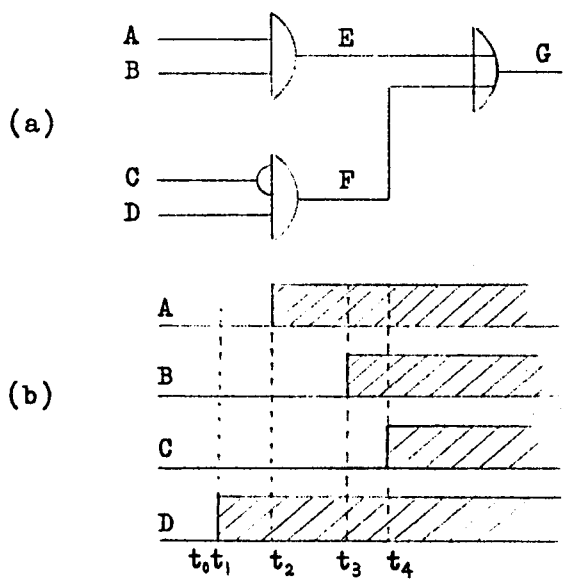


Fig. 4.4 Logical circuit and input pulses.

Fig. 4.4b comes in. Since asynchronous circuit has no timing pulse as synchronous one, the changing points of all the input pulses such as t_1, t_2, t_3, t_4 in Fig. 4.4b, may be corresponded to the clock pulse, and at each such point the states of the logical circuits are calculated.

The simulation program must be written following the flow of signals in the circuits. So the instruction for the OR gate of Fig. 4.4a must be written after the instructions of two AND gates which precede the OR gate. The sequence of the two AND gates is optional in this case. The program is,

```
AND δ 2‖ A‖ B‖ E‖
AND δ 2‖ -C‖ D‖ F‖
OR  δ 2‖ E‖ F‖ G‖
```

When it is desired to print out the output G of the circuit which works by the input data shown in Fig. 4.4b, the program may be as follows.

```
L δ L‖ READ δ 4‖ A‖ B‖ C‖ D‖
       AND δ 2‖ A‖ B‖ E‖
       AND δ 2‖ -C‖ D‖ F‖
        OR δ 2‖ E‖ F‖ G‖
      WRIT δ G‖
        GO δ L‖
```

The corresponding data tape, referring to Fig. 4.4b, is as follows.

```
0000 ‖ 0001 ‖ 1001 ‖ 1101 ‖ 1111 ‖
```

The calculation is done each time a block of data separated by ‖ is read in. However this is an extremely simple

example. An intrinsic property of asynchronous circuit is that, by the change of an input the circuit passes through many internal states which are not stable and finally reaches to the stable state if it exists. By the above simple calculation this phenomenon does not occur. To know the circuit has reached to the final state, the state of the circuit is repeatedly calculated by the input unaltered. If the state of the circuit does not change by the repeated calculation, it has reached the final stable state.

Let us represent an asynchronous circuit by the functions

$$\begin{aligned} Y_1 &= f_1 (X_1, X_2, \cdots, X_n, Y_1, Y_2, \cdots, Y_m) \\ Y_2 &= f_2 (X_1, X_2, \cdots, X_n, Y_1, Y_2, \cdots, Y_m) \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ Y_m &= f_m (X_1, X_2, \cdots, X_n, Y_1, Y_2, \cdots, Y_m) \end{aligned} \qquad (1)$$

where $X_1, X_2, \cdots, X_n$ are inputs to the circuit, and $Y_1, Y_2, \cdots, Y_m$ are the elements representing the internal state of the circuit. These equations mean that the new internal state $(Y_1, Y_2, \cdots, Y_m)$ is calculated by the inputs $X_1, X_2, \cdots, X_n$ and the internal state at this instance. Therefore if the new internal state is represented by $(Y_1^*, Y_2^*, \cdots, Y_m^*)$, it satisfies the following equations.

$$\begin{aligned} Y_1^* &= f_1 (X_1, X_2, \cdots, X_n, Y_1, Y_2, \cdots, Y_m) \\ Y_2^* &= f_2 (X_1, X_2, \cdots, X_n, Y_1, Y_2, \cdots, Y_m) \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ Y_m^* &= f_m (X_1, X_2, \cdots, X_n, Y_1, Y_2, \cdots, Y_m) \end{aligned} \qquad (2)$$

If $(Y_1^*, Y_2^*, \cdots, Y_m^*)$ is equal to $(Y_1, Y_2, \cdots, Y_m)$, the circuit has reached to the final stable state. If not the fol-

lowing substitution is done,

$$Y_1^* \rightarrow Y_1$$
$$Y_2^* \rightarrow Y_2$$
$$\cdots\cdots \qquad (3)$$
$$Y_m^* \rightarrow Y_m$$

and the equations (2) are calculated until finally Yi* is equal to Yi for all i. The operation of (3) can be done by

DEL δ 1 ‖ Yi* ‖ Yi ‖

In actual circuits the delay time required for the operation (3) is different for all Yi, which is the cause of the "race" phenomenon. If the circuit is constructed to have no race phenomenon or to have the same final state even if the transient path is different by race, the above treatment is enough. If this is not the case, the order of calculation of the equations (2) has the effect to the final result.

For example let us consider the circuit of Fig. 4.5.

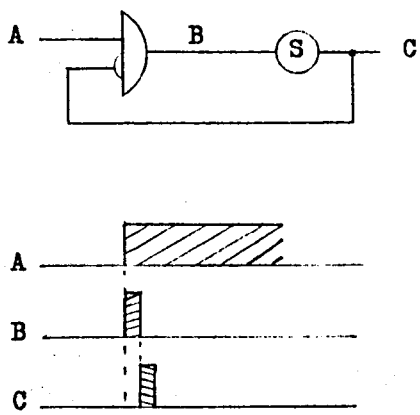


Fig. 4.5 A feedback circuit with a slight delay time.

Supposing that the initial state of the circuit is zero, the input is changed from zero to one. Then B and C become one

for a moment but again return to zero. This is because the circuit S has a slight delay time. If this delay time is taken as a unit of time, the simulation program can be written in the following way.

```
          DEF δ C ‖ O ‖
L δ L1 ‖ READ δ 1 ‖ A ‖
L δ L2 ‖ AND δ 2 ‖ A ‖ -C ‖ B ‖
          TEQ δ B ‖ C ‖ L3 ‖
          WRIT δ C ‖
          GO δ L1 ‖
L δ L3 ‖ DEL δ 1 ‖ B ‖ C ‖
          GO δ L2 ‖
```

and data tape is

```
O ‖ 1 ‖ .
```

4.3.2 Synchronous Circuits

Definite timing signal is used for synchronous circuits, and it is only necessary to consider the circuit operation during the existence of this clock pulse. The unit delay time is corresponded to the interval of the clock pulses. For example for the circuit shown in Fig. 4.6 if the circuits C and

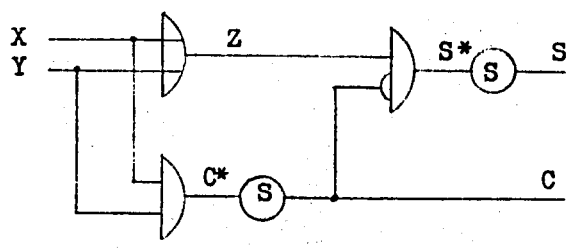


Fig. 4.6 An example of a logical circuit.

S work synchronously with the clock pulse, the circuit operation can be written in the following way.

```
L δ L ‖  READ δ 2 ‖ X ‖ Y ‖
         OR  δ 2 ‖ X ‖ Y ‖ Z ‖
         AND δ 2 ‖ X ‖ Y ‖ C* ‖
         AND δ 2 ‖ Z ‖ -C ‖ S* ‖
         DEL δ 1 ‖ S* ‖ S ‖
         DEL δ 1 ‖ C* ‖ C ‖
         WRIT δ S ‖ C ‖
         GO  δ L ‖
```

There is no need to compare S* with S and C* with C in the synchronous circuit. The calculation from top to bottom corresponds to one bit time operation.

4.4 Assembler

The storage of KDC-I is a drum of 4200 words. In order to accept as long a program as possible, the assembler is constructed to punch out the compiled result when a pseudo instruction is read in. In this way the storage size during the assemblage can be made small. The assembler is a 550 words program while its working storages are prepared from the address 4199 in descending order. When a label is read in, the assembler searches whether the symbol is already memorized and is assigned an absolute address in the table or not. If not, it is memorized and is given an absolute address.

Since the assembler punches out the machine instructions every time when a pseudo instruction is read in, there arises

a trouble that the machine instructions can not be constructed for GO, IF, TEQ and so on whose destination of the jump is not read in still at that stage. Therefore to save this the jump operation is done indirectly by the reference of the jump table, in which the corresponding absolute address is stored. The storage allocation is shown in Fig. 4.7.

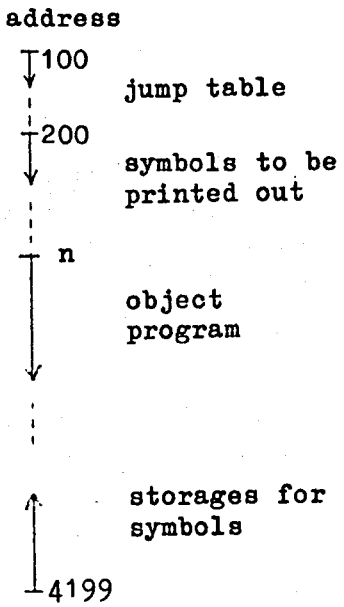


Fig. 4.7 Storage allocation of the programming system.

4.5 Compiler of Boolean Expressions

The pseudo instructions explained in 4.2 are convenient for the logical circuits which are drawn in the chart of AND gate, OR gate and so on. However if such combinatorial logical circuit is complicated, it is troublesome to write pseudo instructions. So a compiler is constructed which accepts logical functions and changes them into machine instructions.

The acceptable logical functions must be of the following form. The definition is by Backus normal form, which is used in defining ALGOL 60.

<Boolean primary>:: = <Boolean variable> | (<Boolean expression>)

<Boolean secondary>:: = <Boolean primary>|- <Boolean primary>

<Boolean factor>:: = <Boolean secondary> | <Boolean factor> * <Boolean secondary>

<Boolean expression>:: = <Boolean factor> | <Boolean expression> + <Boolean factor>

<Logical function>:: = LOG δ<Boolean variable> = <Boolean expression>/

Boolean variable is the same as labels explained in 4.2. This is an operator grammar studied in Chapter 2, PART I. The precedence matrix for this grammar is shown in Fig. 4.8. The flow chart of the compiler of this logical function is shown in Fig. 4.9a--e. S[i] is used as push-down pop-up memory. The contents inside the bracket [ ] are the machine instructions to be generated.

The logical function of Fig. 4.4 can be written as,

LOG δ G = A*B + - C*D/

The corresponding machine instructions generated by the compiler are as follows.

```
ADD/    A
AND     B
STO     #WO
```

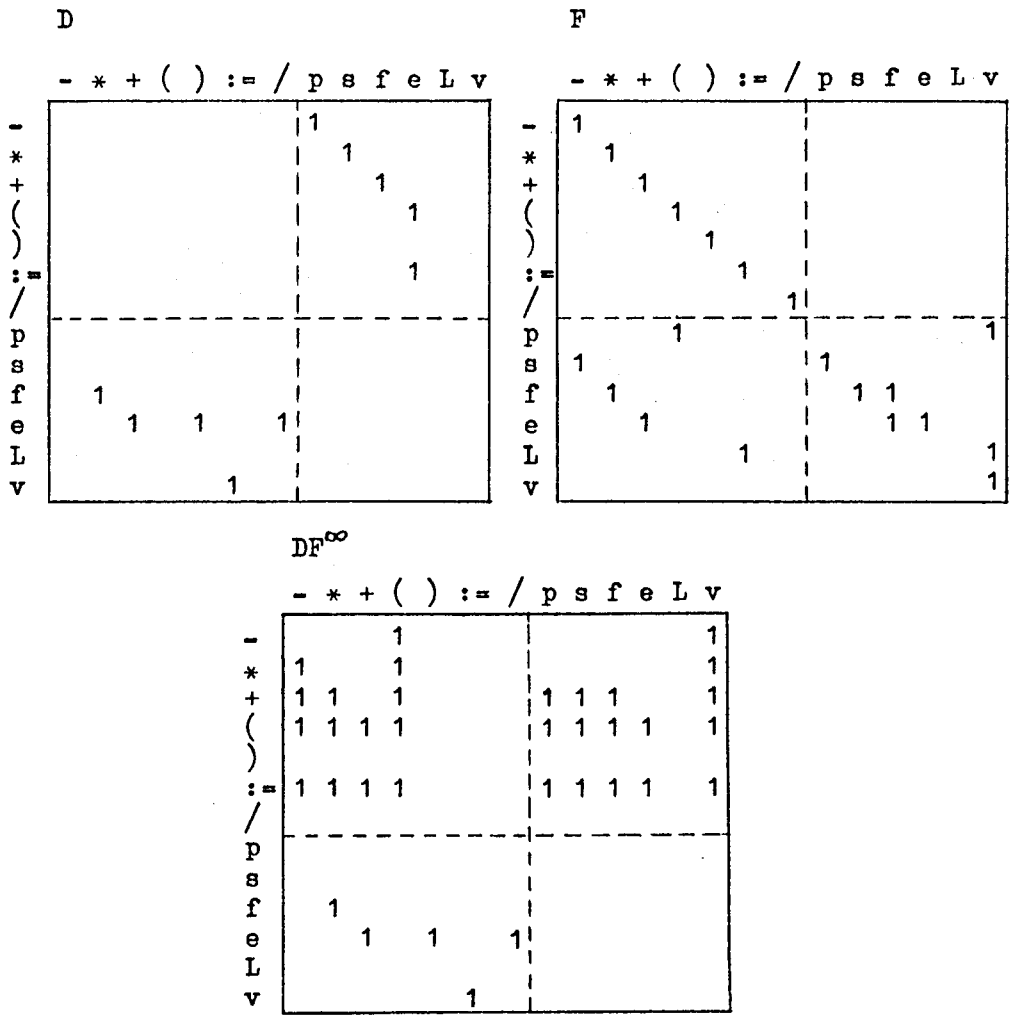


Fig. 4.8 Precedence matrix DF^{∞} with matrices D and F. p: Boolean primary, s: Boolean secondary, f: Boolean factor, e: Boolean expression, L: Logical function, v: Boolean variable. Symbols for column have priority to symbols for row, where 1 exists in the precedence matrix.

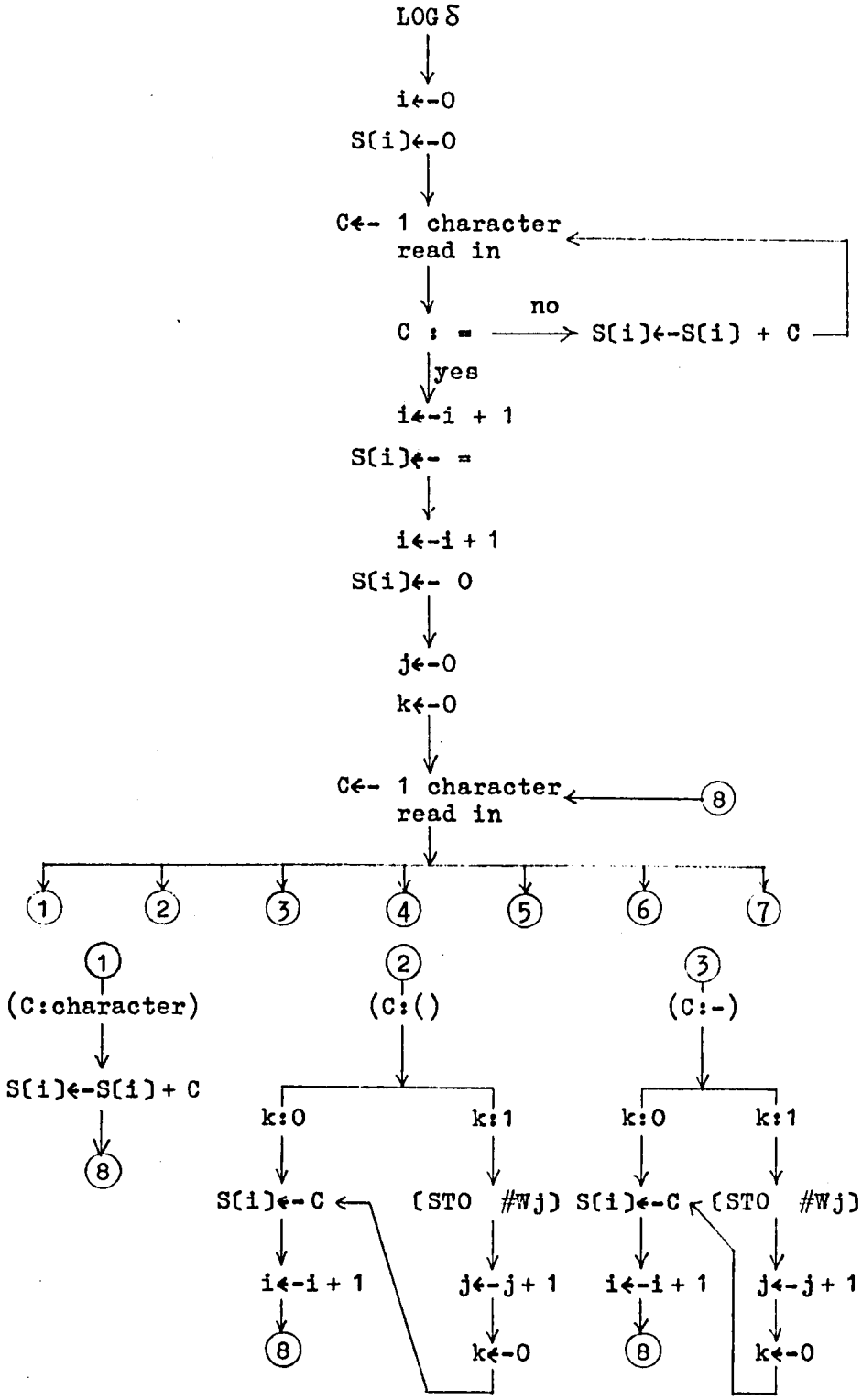


Fig. 4.9a Flow chart of a compiler of Boolean expression.

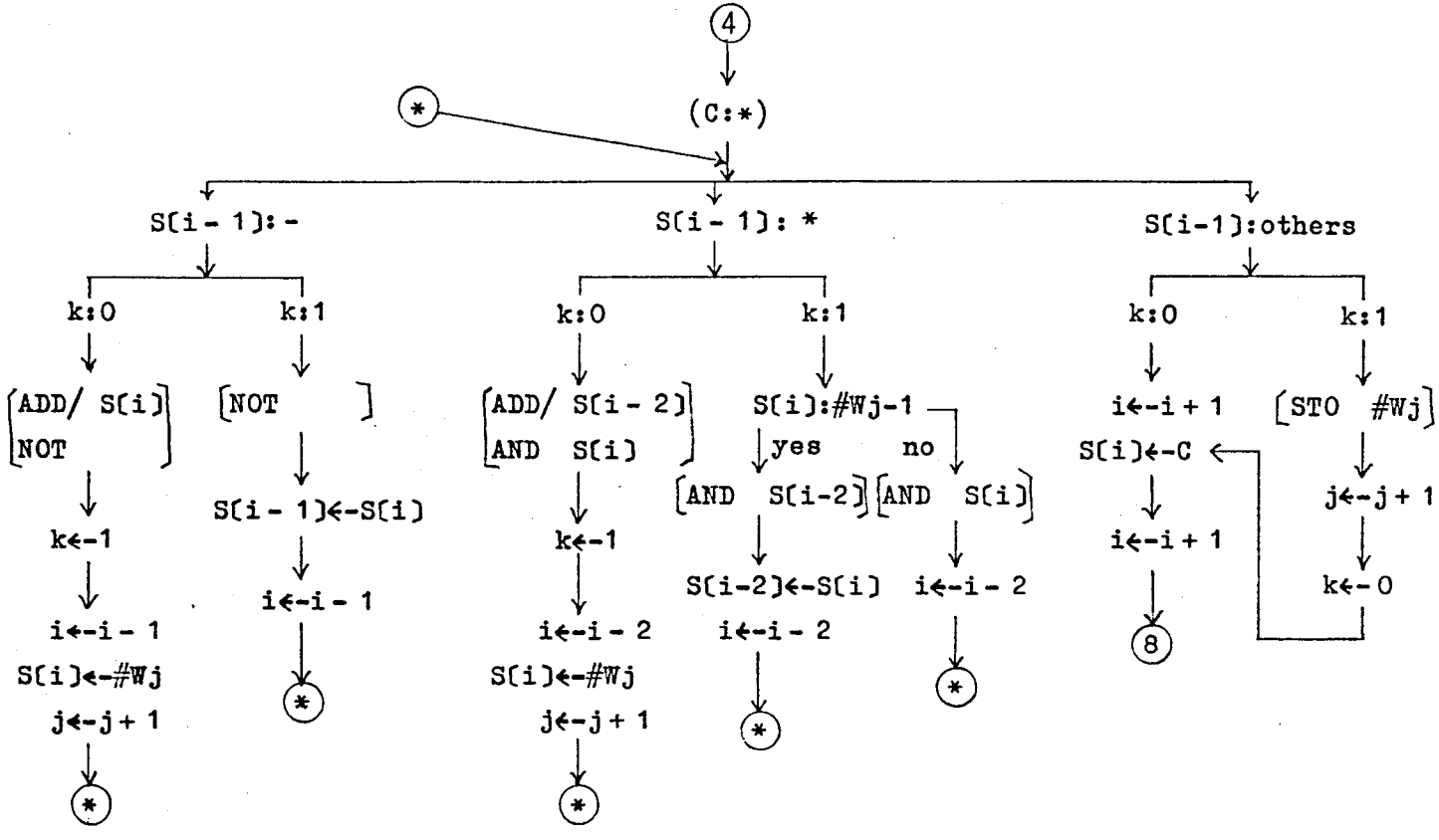


Fig. 4.9b Flow chart of a compiler of Boolean expression. (continued)

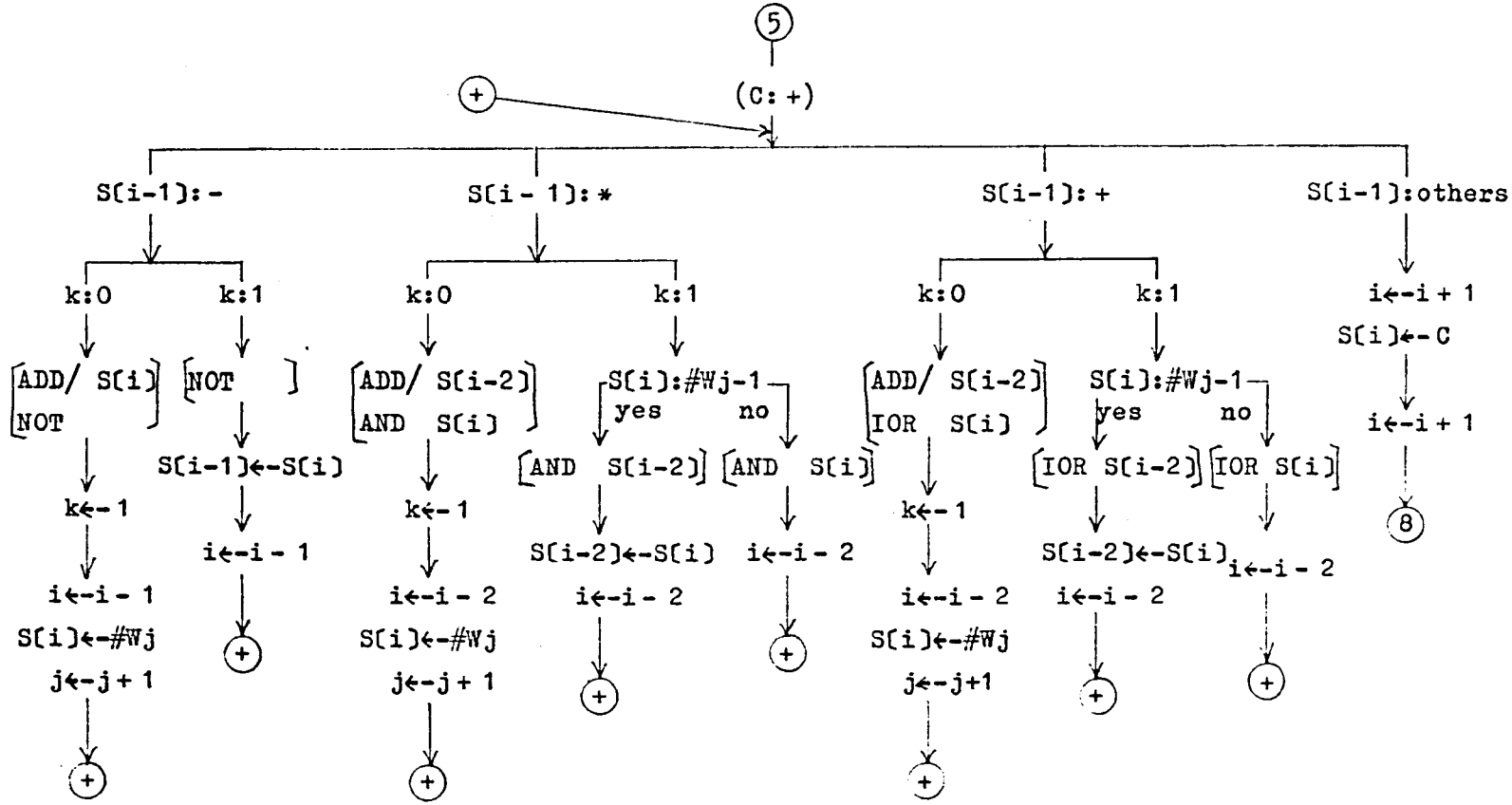


Fig. 4.9c Flow chart of a compiler of Boolean expression. (continued)

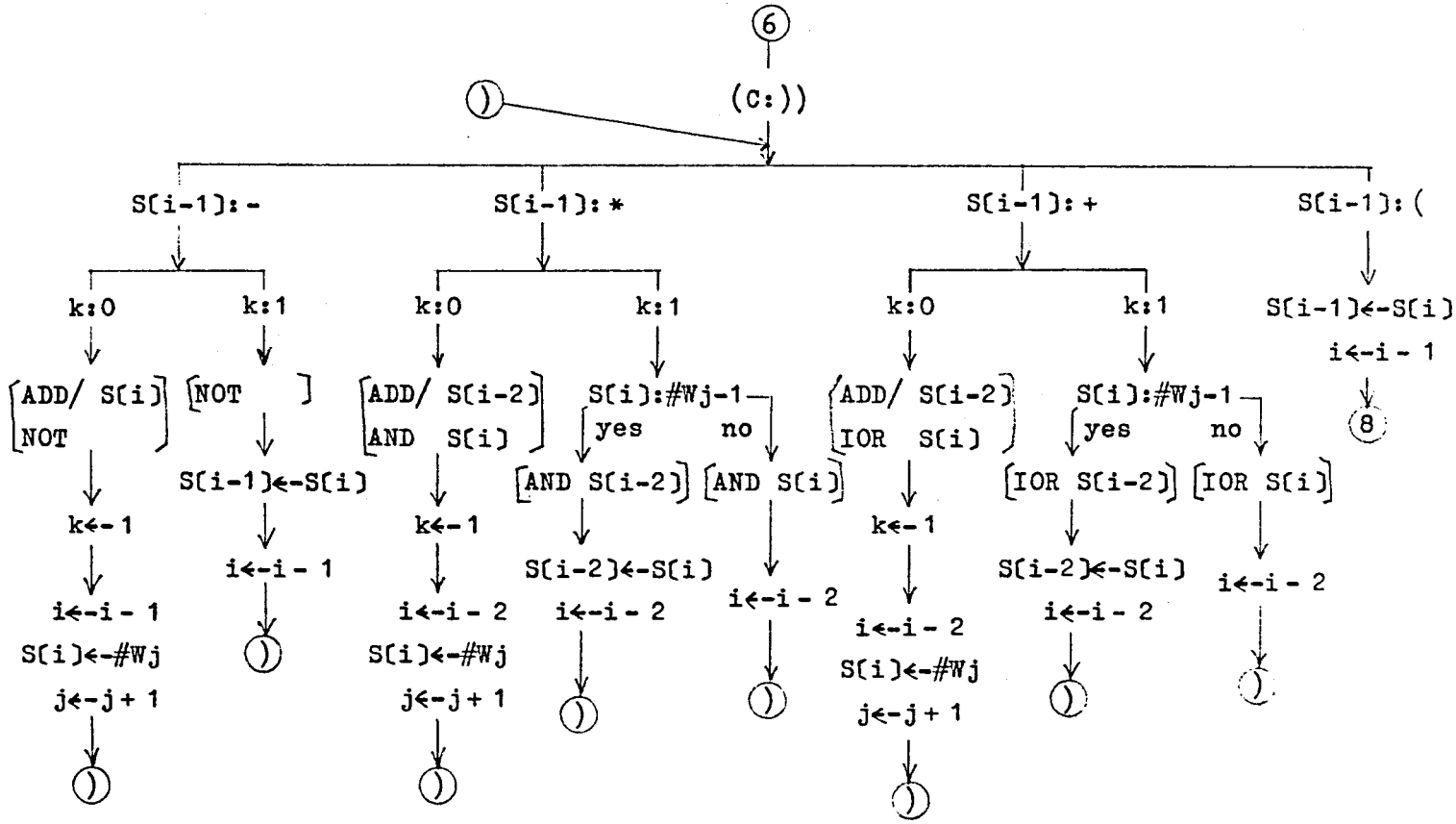


Fig. 4.9d Flow chart of a compiler of Boolean expression. (continued)

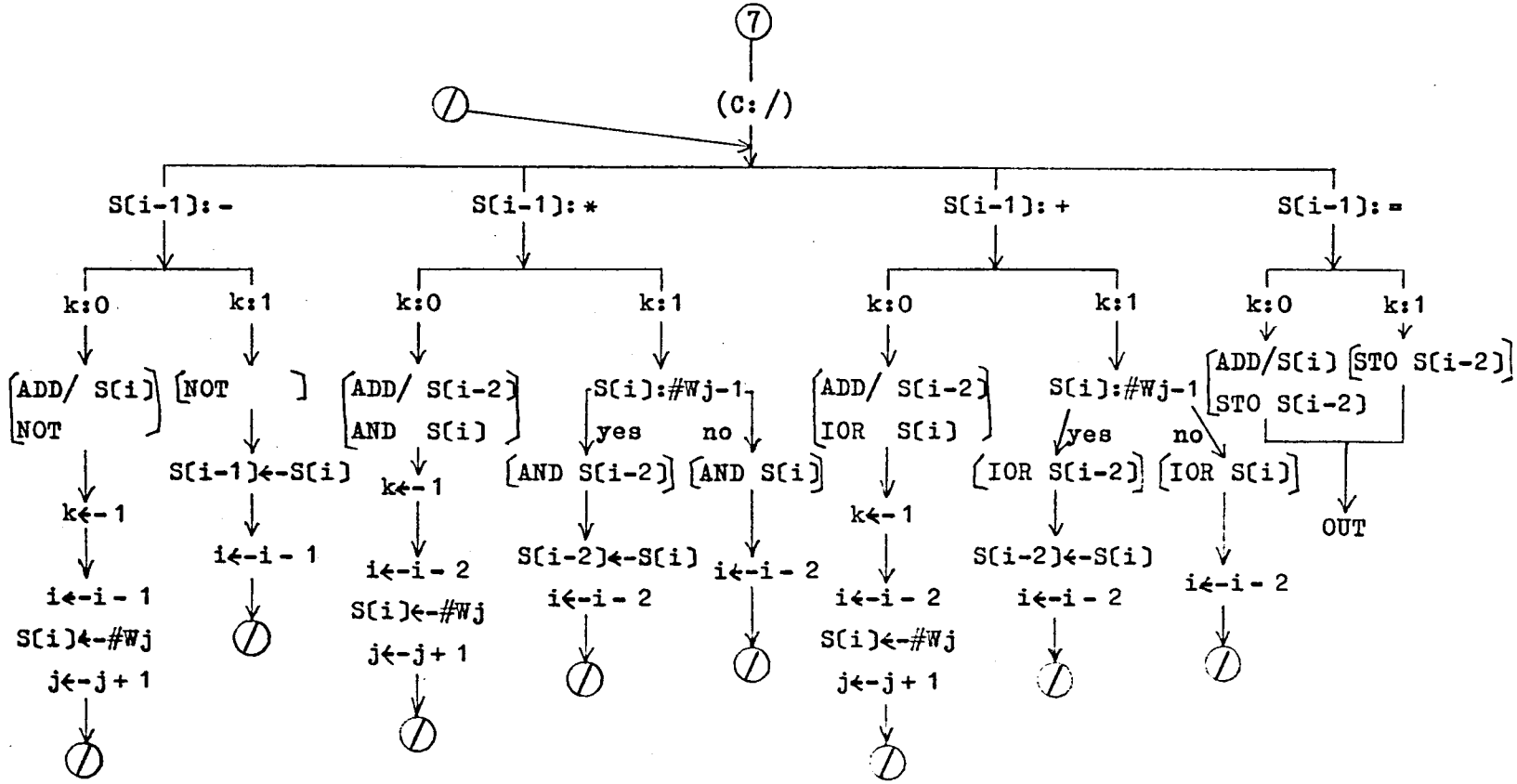


Fig. 4.9e Flow chart of a compiler of Boolean expression. (continued)

```
ADD/     C
NOT
AND      D
IOR      #WO
STO      G
```

4.6 Simulation of Character Recognition Machines

This programming system was used for the simulation of character recognition machines explained in Chapters 2 and 3.3. The main object of the simulation of the machine explained in Chapter 2 was to find out efficient characteristic feature logics. When the number of characters to be recognized increases, it is very difficult to know whether certain logics or decision mechanisms, which are sure to work very well for the very restricted local situations or environment, have no contradiction in the performance of overall system or not. Characteristic feature logics were improved many times by the results of simulation of the system. More than 200 digitized patterns were used for the samples of the simulation. The simulated logical circuits were composed of about 900 transistors, and had about 75 memory circuits of 4 internal states. The assembled machine instructions were about 2000 words and the time necessary for the assemblage was about 90 minutes. Each time a label is read in, it is searched in the label table whether it already exists in the table or not. A fairly long time was necessary for this table search when the size of the label table becomes more than 100. Also the punching out

of the assembled machine instructions required a long time.

Generally there are two kinds of objects in the simulation experiments by computer. One is to find out design errors of a system, and the other is to determine the optimum design for a system. An example of the former is to find out errors in the logical design of decimal adder and so on, in which case the system is to produce fixed outputs for the corresponding fixed inputs. Almost all the simulations have been done for this purpose.

An example of the latter case is a simulation of a character recognition machine, in which case the weight of the simulation is to find out efficient characteristic feature logics rather than the error finding of the system. The simulation in this sense is more important because it plays an essential role in the design of a system.

However input data for the system to be simulated is to be selected very carefully so that the data well represent the simulation purposes. For example, for the simulation of the asynchronous character recognition machine, the resolution of the mesh for the digitization of input pattern should have been determined very carefully. The asynchronous system works even for such very short pulses, that the effect of the noise can not be known if the sampling is very rough. Actually the irregularity of the edge of the vertical line was not represented enough in the digitized patterns, so that some special phenomena relating this could not have been found until the machine was actually constructed.

CHAPTER 5

CONCLUSION

There seems to exist a considerable gap between the theories of pattern recognition and the actual recognition machines. There are many problems in the construction of a machine, which do not arise or are not noticed in the theoretical studies. Here the essential point of the system's design consists in. By the asynchronous character recognition machine explained in this thesis are shown several of these problems very definitely.

Considering the machine as an experimental one, the system is satisfactory, although there are many parts to be improved or to be added for the commercial use. The resolution of a pattern into 16 vertical channels were just satisfactory for numerals and capital English alphabet. Channel selection circuits worked fairly well for lines of characters which are exactly horizontal. However this is a rather severe condition for the commercial use. The method explained in 2.3.3, which determines the position of every character, is to be implemented. If the budget permits the realization of this circuit, it is sure to contribute very much to the improvement of the machine.

Characteristic feature logics are to be examined once more. In 2.2.2 a reason is presented not to minimize the logical functions. But for the commercial machine where characters to be recognized are fixed, the minimization may be useful. When input patterns are too deteriorated the method is

not good even if each characteristic feature logics are taken very marginally. In this case a method is to be found which determines the most likely character among many candidates. Likelihood ratio test is to be realized in circuit in the form compartible with the asynchronous recognition method here explained.

Three kinds of asynchronous recognition circuits are explained and all are actually used properly for certain characters. For the recognition of small English alphabet or for the recognition of Japanese KATAKANA letters some more additional recognition principle will be necessary. For the recognition of more complex characters this principle is not enough. In this case digital computer must be used. For the purpose of this computer input the high speed puncher is provided for the output of digitized pattern as explained in 2.4. 5.

For the character recognition machine to be used as computer input device, as a device on tele-communication line, or as check reader the recognition rate must be perhaps more than 99.99%. However this rate is rather difficult to realize without any checking mechanism. There are roughly two kinds of checking. One is to design characters which are very easy for checking, and the other is to add one or two positions as check digits by the reading of a group of characters. An example of the former is CMC 7 as shown in Fig. 2.5b. Among six intervals two are wide, so that this is used as a check of a character. The latter examples are many. Sum check is

most popular. If these check mechanisms can be available 99.99% may be achieved.

The development of document feeder, typing mechanism, and high quality type ribbon is essential for the character recognition system, although here in this thesis they were not studied.

The recognition principle explained in Chapter 2 can be thought as a method of finding out the topological features of lines. For example line figures shown in Fig. 5.1 can be easily detected by setting up several vertical sections along the horizontal axis.

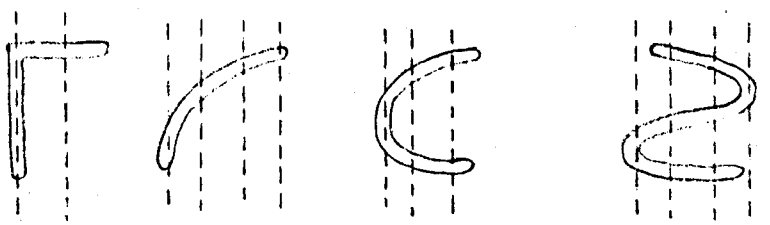


Fig. 5.1 Topological features which can be detected by the principle mentioned in Chapter 2.

A good point of the constructed machine is that the system is very simple and considerably inexpensive. The asynchronous character recognition machine illustrated in 3.3, which has two dimensional input part, is simpler than this, because no memory circuits are necessary. From the simulation experiment this machine is highly recommendable for the commercial use.

A programming language illustrated in Chapter 4 was very convenient for the simulation of asynchronous character rec-

ognition machines presented in this thesis. This kind of programming system is essential for many problems of information processing.

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to Professor Toshiyuki Sakai for his constant guidance and encouragement during the course of this investigation. He is deeply indebted to Professor Takeshi Kiyono for his constant and generous support to the author's study.

He is also very much indebted to Assistant Professor S. Doshita for his suggestions in the construction of the character recognition machine, to Mr. K. Sugata, Mr. Y. Ni-imi, and Mr. Z. Fukuda for their fine cooperations and help in the construction and adjustment of the character recognition machine, to Mr. S. Sugita and Mr. S. Uemura for their cooperations and discussions in the experiments of sentence generation.

He owes a considerable debt of gratitude to the director of the Electronics Mechanics Laboratory Dr. K. Nagamori, Dr. S. Sekiguchi, Mr. K. Kiji, of Nippon Electric Company, Ltd. for their cooperation in the construction of the character recognition machine.

The author wishes to thank the staff and students of Professor T. Sakai's laboratory, especially Mr. H. Nishio, for their daily discussions and help during his study.

BIBLIOGRAPHY

(1) T. Sakai, "Pattern Recognition," J. Information Processing Society of Japan, Vol.3, No.4, July, 1962.

(2) M.E. Stevens, "Automatic Character Recognition: A State-of-the-Art Report," NBS Report 7175-A,B. May, 1961.

(3) G.L. Fischer, Jr. et al., ed. "Optical Character Recognition," Spartan Books, 1962.

(4) L.D. Harmon, "A Line-Drawing Pattern Recognizer," Proc. WJCC, May, 1960.

(5) "Character Set for Optical Character Recognition," Comm. ACM, Vol.8, No.1, Jan., 1965.

(6) ISO/TC 97 (Secretariat-32)65, "First Draft Proposal; Print Specifications for Magnetic Ink Character Recognition," August 27, 1964.

(7) T. Sakai & T. Fukinuki, "Fundamental Design of A Pattern Recognition Machine," Technical Report of the Professional Group on Automaton and Automatic Control of IECEJ*, Jan., 1961.

(8) G.S. Sebestyen, "Decision-Making Processes in Pattern Recognition," The Macmillan Co. New York, 1962.

(9) M.L. Minsky, "Steps Toward Artificial Intelligence," Proc. IRE,pp.8-30, Jan., 1961.

(10) T. Sakai, M. Nagao, et al., "Design of an Asynchronous Character Recognition Machine," Technical Report of the Professional Group on Automaton and Automatic Control of IECEJ*, Jan., 1965, and
T. Sakai, M. Nagao, et al., "An asynchronous character recognition system and device," to appear in the Proc. IFIP Congress 65, Vol.2.

(11) M. Tanaka, "Design of a character recognition machine by asynchronous sequential logic circuit," Bachelors thesis, Kyoto University, 1961.

(12) T. Sakai, M. Nagao & Y. Ni-imi, "A Method of Character

Recognition," Record of the 1963 Joint Convention of the IECEJ* and others.

(13) M. Nagao, "Pattern recognition machine by two dimensional input part," Mimeograph, Sept. 1964.

(14) T. Sakai & M. Nagao, "A programming System for the Simulation of Logical Circuit," Technical Report of the Professional Group on Automaton and Automatic Control of IECEJ*, Jan., 1964.

(15) M. Nagao, "Simulation of Logical Circuit and Design of Character Recognition Machine," Record of the 1964 Joint Convention of the Kansai District of IECEJ* and others, S6-4, 1964.

(16) K. Takashima, et al, "Logical Construction Simulation Program," Technical Report of the Professional Group on Electronic Computer of IECEJ*, Oct., 1962.

(17) H.M. Markowitz, et al., "SIMSCRIPT, A Simulation Programming Language," Prentice Hall, 1963.

(18) T. Kiyono, et al., "KDC-I Instruction (KDC-I Manual No.1)" Kyoto Univ. Computer Center, 1959.

(19) S. Yajima, "Studies on the Design and Construction of an Electronic Digital Computer," Doctral thesis, Kyoto University, July 1963.

(20) Nippon Electric Co. "Manual of GC-130 Magnetic Core Memory," Tokyo, 1964.

*IECEJ : The Institute of Electrical Communication Engineers of Japan.

APPENDIX

TABLE 1. KDC-I INSTRUCTIONS

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
100	ADD I J A	0.50+Ai+Aa	Add. c(AC)pUA+c(#IJA)→c(AC).
102	ADA I J A	0.50+Ai+Aa	Add Absolute. c(AC)pUA+|c(#IJA)|→c(AC).
104	SUB I J A	0.50+Ai+Aa	Subtract. c(AC)pUA-c(#IJA)→c(AC).
106	SBA I J A	0.50+Ai+Aa	Subtract Absolute. c(AC)pUA−|c(#IJA)|→c(AC).
110	ADM · · −	0.50+Ai	Add MD. c(AC)pUA+c(MD)→c(AC).
114	SBM · · −	0.50+Ai	Subtract MD. c(AC)pUA-c(MD)→c(AC).
120	MPA I J A	5.8 av+Ai+Aa	Multiply and Add. c(AC)+c(MD)•c(#IJA)→c(AC).
122	MPS I J A	5.8 av+Ai+Aa	Multiply and Subtract. c(AC)-c(MD)•c(#IJA)→c(AC).
130	RAA I J n	0.50+Ai	Raise Address. c(AC)pUAad+#IJn→c(AC).
134	LWA I J n	0.50+Ai	Lower Address. c(AC)pUAad-#IJn→c(AC).
138	RND I J n	0.50+Ai	Round. c(AC)rnd→c(AC).
140	ADR I J A	6.9 av+Ai+Aa	Add, Divide and Round or Halt. [{c(AC)pUA+c(#IJA)}/c(MD)]rnd→c(UA); 0→c(LA), or Add and Halt.
150	DVJ I J A	6.0 av+Ai	Divide or Jump. c(AC)/c(MD)→c(UA); Rem→c(LA), or Jump, [R].
152	DRJ I J A	6.6 av+Ai	Divide and Round or Jump. {c(AC)/c(MD)}rnd→c(UA); 0→c(LA), or Jump.
160	ADL I J A	0.55+Ai+Aa	Add to LA. c(AC)pLA+c(#IJA)→c(AC).
162	AAL I J A	0.55+Ai+Aa	Add Absolute to LA. c(AC)pLA+|c(#IJA)|→c(AC).
164	SBL I J A	0.55+Ai+Aa	Subtract from LA. c(AC)pLA-c(#IJA)→c(AC).
166	SAL I J A	0.55+Ai+Aa	Subtract Absolute from LA. c(AC)pLA-|c(#IJA)|→c(AC).
200	FAD I J A	1.3+Ai+Aa	Floating Add. c(AC)+c(#IJA)→c(AC).
202	FAA I J A	1.3+Ai+Aa	Floating Add Absolute. c(AC)+|c(#IJA)|→c(AC).
204	FSB I J A	1.3+Ai+Aa	Floating Subtract. c(AC)−c(#IJA)→c(AC).
206	FSA I J A	1.3+Ai+Aa	Floating Subtract Absolute. c(AC)-|c(#IJA)|→c(AC).
210	FAM · · −	1.3+Ai	Floating Add MD. c(AC)+c(MD)→c(AC).
214	FSM · · −	1.3+Ai	Floating Subtract MD. c(AC)-c(MD)→c(AC).
221	FMP/ I J A	5.2 av+Ai+Aa	Clear and Floating Multiply. c(MD)•c(#IJA)→c(AC).

TABLE 1. (Continued)

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
223	FMC/ I J A	5.2 av + Ai + Aa	Clear, Floating Multiply and Change Sign. -c(MD)*c(#IJA) → c(AC).
238	FRD I J n	0.55 + Ai	Floating Round. c(AC)rnd → c(AC).
240	FAV I J A	6.7 av + Ai + Aa	Floating Add, Divide and Round or Halt. [{c(AC) + c(#IJA)}/c(MD)]rnd → c(UA); 0 → c(LA), or Add and Halt.
250	FDJ I J A	5.1 av + Ai	Floating Divide or Jump. c(AC)/c(MD) → c(UA); Rem → c(LA), or Jump, [R].
252	FDR I J A	5.6 av + Ai	Floating Divide and Round or Jump. {c(AC)/c(MD)}rnd → c(UA); 0 → c(LA), or Jump.
300	STO I J A	0.35 + Ai + Aa	Store. c(UA) → c(#IJA).
302	SLA I J A	0.35 + Ai + Aa	Store LA. c(LA) → c(#IJA), N.B. R-ind.
304	STM I J A	0.35 + Ai + Aa	Store MD. c(MD) → c(#IJA).
306	STA I J A	0.35 + Ai + Aa	Store Address. c(UA)ad → c(#IJA)ad.
308	STL I J A	0.35 + Ai + Aa	Store Location. c(LC) → c(#IJA)ad.
310	PAM • • -	0.35 + Ai	Place UA to MD. c(UA) → c(MD).
320	LDM I J A	0.35 + Ai + Aa	Load MD. c(#IJA) → c(MD).
322	LDA I J A	0.35 + Ai + Aa	Load Address. c(#IJA)ad → c(UA)ad.
324	CMP I J A	0.55 + Ai + Aa	Compare. c(LC) + 1, 2, 3 → c(LC), if c(MD) ≷ c(#IJA).
340	FSL I J A	0.95av + Ai + Aa	Floating Store LA. c(LA) → c(#IJA).
360	TLU MJ A	5.5 av + Ai + Aa	Table Look Up. loc(c(MD)) → c(OR)ad, [P], or c(M) → c(LC).
364	LDQ QJ A	2.9 + Ai + Aa	Load Quick Access. c(#JA), ··· → c(Q#JA), ···
366	STQ QJ A	2.9 + Ai + Aa	Store Quick Access. c(Q#JA), ··· → c(#JA), ···
410	FFL • • -	0.80 + Ai	Fixed to Floating. fx(c(AC)) to fl(c(AC)).
450	FFX I J A	0.55 + Ai	Floating to Fixed or Jump. fl(c(AC)) to fx(c(AC)), or Jump.
500	EAD I J A	0.55 + Ai + Aa	Extract and Add. c(AC)pUA + c(#IJA) & c(MD)odd → c(AC).
502	ERE I J A	0.35 + Ai + Aa	Extract and Replace. c(UA) & c(MD)even U c(#IJA) & c(MD)odd → c(UA).
510	SSP • • -	0.30 + Ai	Set Sign Plus. |c(AC)| → c(AC).
512	CHS • • -	0.30 + Ai	Change Sign. -c(AC) → c(AC).

TABLE 1. (Continued)

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
514	**NOP** · · –	0.30 + Ai	No Operation.
516	**NOT** · · –	0.35 + Ai	NOT. $\overline{c(UA)}$ & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA); 0 → c(UA, 8).
518	**SCT** · · –	1.4 av + Ai	Shift and Count. LLS until 1st non-zero appears in UAm, and shift count → c(OR)ad, [P].
520	**AND** I J A	0.35 + Ai + Aa	AND. {c(UA) & c(#IJA)} & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA); 0 → c(UA, 8).
522	**IOR** I J A	0.35 + Ai + Aa	Inclusive OR. {c(UA) U c(#IJA)} & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA); 0 → c(UA, 8).
530	**SLS** I J n	0.55 + Ai	Short Left Shift. left shift of c(UA) by $(\#IJn)_{2,1}$ places.
532	**LLS** I J n	0.55 + Ai	Long Left Shift. left shift of c(AC) by $(\#IJn)_{2,1}$ places.
534	**LCS** I J n	0.55 + Ai	Long Cyclic Shift. cyclic left shift of c(AC) by $c(\#IJn)_{2,1}$ places.
536	**SRS** I J n	0.55 + Ai	Short Right Shift. right shift of c(UA) by $(\#IJn)_{2,1}$ places.
538	**LRS** I J n	0.55 + Ai	Long Right Shift. right shift of c(AC) by $(\#IJn)_{2,1}$ places.
550	**WAN** I J n	0.35 + Ai	Weighted AND. c(UA, 1) & c(UA, #IJn) → c(UA, 1); 0 → c(UA, 2, 4, 8).
552	**WOR** I J n	0.35 + Ai	Weighted OR. c(UA, 1) U c(UA, #IJn) → c(UA, 1); 0 → c(UA, 2, 4, 8).
630	**SEL** I J n	0.35 + Ai	Select Component. select I/O Component specified by #IJn.
632	**RIN** I J n	PTR; 200 ch/s	Read-in. read $(\#IJn)_{2,1}$ char. into c(UA) in mode specified by $(\#IJn)_4$.
634	**WRT** I J n	0.35 + Ai; 8 ch/s	Write. print and/or punch $(\#IJn)_{2,1}$ char. from c(UA) in mode specified by $(\#IJn)_4$.
636	**WSP** I J m	0.35 + Ai; 8 ch/s	Write Special. print and/or punch a char. specified by $(\#IJm)_{4,3}$ $(\#IJm)_{2,1}$ times.
638	**FWR** · · –	0.35 + Ai; 8 ch/s	Floating Write. print and/or punch fl(c(UA)).
710	**HJM** I J A	0.35 + Ai	Halt and Jump. halt and #IJA → c(LC).
712	**JSW** S J A	0.35 + Ai	Jump by Switch. #JA → c(LC), if Switch-S is on.
714	**JMP** I J A	0.35 + Ai	Jump. #IJA → c(LC).
750	**JMI** I J A	0.35 + Ai	Jump on Minus. #IJA → c(LC), if c(AC) <-0.
752	**JUN** I J A	0.35 + Ai	Jump on UA No Zero. #IJA → c(LC), if c(UA) $\neq 0$.
754	**JNZ** I J A	0.35 + Ai	Jump on No Zero. #IJA → c(LC), if c(AC) $\neq 0$.

TABLE 1. (Continued)

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
756	JOV IJ A	0.35+Ai	Jump on Overflow. #IJA → c(LC), if c(UA)v ≠ 0.
758	JEO IJ A	0.35+Ai	Jump on Exponent Overflow. #IJA → c(LC), if c(UA)v ≥ 2.
820	LXA HJ A	0.35+Ai+Aa	Load Index from Address. c(#JA)ad → c(H).
822	STX HJ A	0.35+Ai+Aa	Store Index. c(H) → c(#JA)ad.
830	SEX HJ n	0.35+Ai	Set Index. #Jn → c(H).
832	RAX HJ n	0.35+Ai	Raise Index. c(H)+#Jn → c(H).
834	LWX HJ n	0.35+Ai	Lower Index. c(H)−#Jn → c(H).
850	JXL HJ A	0.35+Ai	Jump with Index Lowered. if c(H) ≠ 0, #JA → c(LC), and c(H)−1 → c(H), or if c(H) = 0, normal seq., and 9,999 → c(H).
852	JXR HJ A	0.35+Ai	Jump with Index Raised. if c(H) ≠ 0, #JA → c(LC), and c(H)+1 → c(H), or if c(H) = 0, normal seq., and 1 → c(H).
854	JXU HJ A	0.35+Ai	Jump with Index Unequal. if c(H) ≠ c(IRI), #JA → c(LC), and c(H)+1 → c(H), or if c(H) = c(IRI), normal seq., and c(H)+1 → c(H).
856	JSX HJ A	0.35+Ai	Jump after Set Index from LC. c(LC) → c(H), and #JA → c(LC).
860	PSX IJ A	0.35+Ai+Aa	Set Pseudo Index. c(#IJA)ad → c(OR)ad, [P].
910	BTP NJ A	0.45+Ai; 220	Buffer to Tape. c(CB) and #JA → c(MT, N; #JA)
912	TPB NJ A	17.0+Ai; 220	Tape to Buffer. c(MT, N) → c(CB); if blook No. = #JA, skip next instruction, [TC].
914	BLS NJ A	0.45+Ai; 100*n+120	Blook Search. c(MT, N; #JA) → c(CB), [TC].
920	DMB ·J A	2.90+Ai+Aa	Drum to Buffer. c(#IJA), ··· → c(4,200), ···
922	BDM ·J A	2.90+Ai+Aa	Buffer to Drum. c(4,200), ··· → c(#IJA), ···
930	RWD N· −	0.45+Ai; 20; av. ca. 450 cm/s	Rewind. rewind (MT, N) to its load point.
932	BST N· −	0.45+Ai; 220	Back Space Tape. test c(MT, N) in backward direction, [TC].
934	TTP N· −	0.45+Ai; 220	Test Tape. test c(MT, N) in forward direction, [TC].
936	ETP N· −	0.45+Ai; 220	Erase Tape. erase one block length of (MT, N) in forward direction.
950	JTG ·J A	0.45+Ai	Jump on Tape Good. if TC-ind. is off, #JA → c(LC), or if on, normal seq.
952	JTE NJ A	0.45+Ai	Jump by Tape End. if No. N TE-ind. is on, #JA → c(LC), or if off, normal seq.

TABLE 1. (Continued)

Notes

The operation time is unchanged whether or not the modification of the address of an instruction takes place. The time for the modification is included in the operation time.
For the instructions which permit concurrent operations, the time needed for the central processing unit is written first, and the time needed for the output unit or the magnetic tape control unit second and so on.

Symbols and Abbreviations

Ai : instruction access time.
Aa : data access time.
→ : the replacement.
: the modification of the address of an instruction by the index registers.
e.g. #IJA ≡ c(I) + c(J) + A (in modulo 10,000).
& : logical AND.
U : logical OR.
— : logical NOT.
rnd : round off.
Rem : remainder.
P-ind. : if the P-ind. is on, the address part of the next instruction in sequence is modified by the result in c(OR)ad of the former instruction, or if off, normal.
R-ind. : remainder indicator.
TC-ind. : magnetic tape check indicator.
[P], [R] or [TC] : the type of an instruction which affects P, R or TC-indicator.
c(X) : the contents of the register X, e.g. c(AC), c(#IJA), etc.
c(I), c(J) or c(H) : the contents of the index register IR I, J or H.
c(Y)ad : the contents of the address part of Y.
c(AC)pUA : the contents of the AC, on the UA part of which the operation is made.
c(UA)v : the content of the 12-th or the overflow digit of the UA.
c(E) & c(MD)odd : the extracted c(E) by the odd numbers in each digit of the MD.
c(CB) : the contents of the core buffer memory.
UAad : the address part of the UA.
UAm : the 11-th or the most significant digit of the UA.
(#IJn)i : the number of the i-th digit of #IJn.
loc(π/2) : the location of a data word whose contents are π/2.
fx(x) : a fixed-point number x, e.g. fx(π).
fl(y) : a floating-point number y, e.g. fl(π).
Q#JA : the address in the quick access band No. Q which satisfies Q#JA ≡ #JA (in modulo 50).
c(UA, i) : the contents of 11 bits specified by the weight i of the UA excluding the UA overflow digit.
(MT, N) : magnetic tape in No. N magnetic tape handler.
c(MT, N) : a block on (MT, N)
c(MT, N; #JA) : a block whose block number is #JA on (MT, N).